Superhypergraph Neural Networks and Plithogenic Graph Neural Networks: Theoretical Foundations, pp. 577-653, in Takaaki Fujita, Florentin Smarandache: A*dvancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond. Fifth volume: Various SuperHyperConcepts (Collected Papers)*. Gallup, NM, United States of America – Guayaquil (Ecuador): NSIA Publishing House, 2025, 653 p.

Chapter 17
Superhypergraph Neural Networks and Plithogenic Graph Neural Networks: Theoretical Foundations

Takaaki Fujita[1]∗, Florentin Smarandache[2],
[1]∗ Independent Researcher, Shinjuku, Shinjuku-ku, Tokyo, Japan. t171d603@gunma-u.ac.jp
[2] University of New Mexico, Gallup Campus, NM 87301, USA. smarand@unm.edu

-

Abstract: Hypergraphs extend traditional graphs by allowing edges to connect multiple nodes, while superhypergraphs further generalize this concept to represent even more complex relationships. Neural networks, inspired by biological systems, are widely used for tasks such as pattern recognition, data classification, and prediction.

Graph Neural Networks (GNNs), a well-established framework, have recently been extended to Hypergraph Neural Networks (HGNNs), with their properties and applications being actively studied. The Plithogenic Graph framework enhances graph representations by integrating multi-valued attributes, as well as membership and contradiction functions, enabling the detailed modeling of complex relationships.

In the context of handling uncertainty, concepts such as Fuzzy Graphs and Neutrosophic Graphs have gained prominence. It is well established that Plithogenic Graphs serve as a generalization of both Fuzzy Graphs and Neutrosophic Graphs. Furthermore, the Fuzzy Graph Neural Network has been proposed and is an active area of research.

This paper establishes the theoretical foundation for the development of SuperHyperGraph Neural Networks (SHGNNs) and Plithogenic Graph Neural Networks, expanding the applicability of neural networks to these advanced graph structures. While mathematical generalizations and proofs are presented, future computational experiments are anticipated.

Keywords: hypergraph, superhypergraph, Neural Network, Neutrosophic Graph, Fuzzy Graph
MSC2010 (Mathematics Subject Classification 2010): 05C65 - Hypergraphs, 05C82 - Graph theory with applications, 03E72 - Fuzzy set theory

1 Introduction

1.1 Hypergraphs and Superhypergraphs

Graph theory, a pivotal area of mathematics, focuses on understanding networks composed of vertices (nodes) and edges (connections)[100, 102]. These mathematical structures effectively model relationships, dependencies, and transitions among elements, making them versatile tools across various domains [45,58,95,156].

The foundational significance of graph theory has spurred its development and application in numerous disciplines, including:

- *Computational Sciences*: Graphs are essential in designing circuits and optimizing computational workflows, as highlighted in recent studies on graph-based optimization techniques [40, 41, 405].
- *Chemistry and Biology*: Chemical graph theory models molecular structures and interactions [42, 380], while bioinformatics leverages graphs to study protein structures and gene interactions [6, 373, 377].
- *Project Management*: Graphs are utilized to analyze workflows and dependencies, facilitating efficient resource allocation and scheduling in project management frameworks [202, 296, 368].
- *Probabilistic Modeling*: Bayesian networks employ graph structures to represent conditional dependencies among random variables [277, 418].
- *Graph Databases*: Modern data storage and retrieval systems increasingly rely on graph databases for their ability to model complex relationships effectively [21, 22, 31, 141, 166, 261, 304].

A hypergraph is a generalization of a conventional graph, extending and abstracting concepts from graph theory [51, 60, 152, 153, 164]. Hypergraphs have wide-ranging applications across fields such as machine learning, biology, social sciences, and graph database analysis, among others (e.g., [69, 85, 139, 187, 232, 403, 427, 443]). From a set-theoretic perspective, a hypergraph can, without risk of misunderstanding, be viewed as the powerset of its vertex set.

The concept of SuperHyperGraph has recently emerged as a more general extension of hypergraphs, generating substantial research interest similar to that seen in the study of hypergraphs[126, 130, 340]. Numerous investigations have been carried out in this field [122, 126, 128, 130, 170, 171, 340, 341, 343, 346, 351].

A Superhypergraph is a type of Superhyperstructure. It can be regarded as an extension of the concept of an n-th-Power Set[331] applied to graphs. The definitions of Superhyperstructure and n-th Power Set are provided below.

Definition 1.1 (n-th powerset)**.** (cf.[331, 352]) The n-th powerset of H, denoted $P_n(H)$, is defined recursively as:

$$P_1(H) = P(H), \quad P_{n+1}(H) = P(P_n(H)) \quad \text{for } n \geq 1.$$

Similarly, the n-th non-empty powerset of H, denoted $P_n^*(H)$, is defined as:

$$P_1^*(H) = P^*(H), \quad P_{n+1}^*(H) = P^*(P_n^*(H)).$$

Definition 1.2. (cf.[331, 352]) A *SuperHyperStructure* is a mathematical structure defined as a pair:

$$\mathcal{S} = (P_n^*(H), O),$$

where:

1. $P_n^*(H)$ is the n-th non-empty powerset of H, which excludes the empty set.
2. O is a set of operations or relations, called *SuperHyperOperators*, defined on $P_n^*(H)$.

Example 1.3 (Example of SuperHyperOperators)**.** (cf.[331, 352]) A binary SuperHyperOperator $\circ$ can be defined as:

$$\circ : P_n^*(H) \times P_n^*(H) \to P_n^*(H).$$

For example, given two elements $A, B \in P_n^*(H)$, their operation under $\circ$ might be defined as:

$$A \circ B = \{C \mid C = f(A, B) \text{ for some function } f\}.$$

Other examples of Superhyperstructures include Superhyperalgebras[197, 198, 212, 213, 221, 299, 300, 331, 342], Superhypertopology[348, 349, 358, 407, 422], Superhyperfunctions[345, 350], and Superhypersoft sets[126, 127, 265, 347, 360], all of which are well-known in this field. Therefore, research on hypergraphs and superhypergraphs is significant from both mathematical and practical perspectives.

For reference, the relationships between Superhypergraphs are illustrated in Figure 1.

1.2 Graph Neural Networks

This subsection provides an overview of Graph Neural Networks. In recent years, fields such as machine learning (cf. [28, 186, 273, 304, 405, 419]), artificial intelligence (cf. [5, 34, 321, 374]), and big data (cf. [49, 79, 200, 257]) have gained significant prominence. This paper focuses on neural networks, which play a pivotal role in these domains.

A neural network is a computational model inspired by biological neural systems, designed for tasks such as pattern recognition, data classification, and prediction [20, 25, 46, 223, 393, 411, 412]. Building upon this foundation, a Graph Neural Network (GNN) extends neural networks to graph structures, enabling the modeling of relationships between nodes, edges, and their associated features [94, 205, 269, 297, 316, 324, 386, 404, 429, 440, 447].

Building on this concept, Hypergraph Neural Networks (HGNNs) extend traditional Graph Neural Networks (GNNs) by leveraging hyperedges to capture higher-order relationships that involve multiple nodes simultaneously [70, 115, 181, 183, 204, 369, 401]. Related concepts include Hypernetworks, which have been studied extensively in works such as [76, 167, 225, 363, 388]. Additionally, networks built on directed graphs, such as Directed Graph Neural Networks [177–179, 325, 450], and those based on mixed graph structures, such as Mixed Graph Neural Networks [163], are also well-known.

Given the wide range of applications studied in these areas, research into Graph Neural Networks is of critical importance.

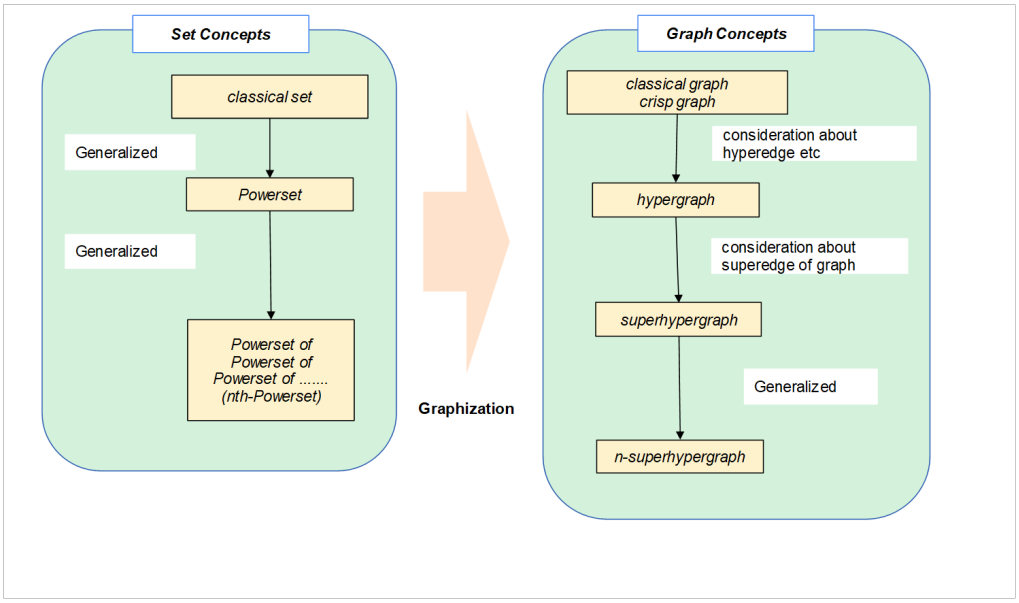

Fig. 1. Some Superhypergraphs Hierarchy.

1.3 Uncertain graphs

The concept of fuzzy sets was introduced in 1965 [430]. Fuzzy sets provide a framework for addressing uncertainty in the real world and have been applied in various fields, including graph theory, algebra, topology, and logic. Furthermore, extensions of fuzzy sets, such as neutrosophic sets [332, 334], have been developed to handle even more complex forms of uncertainty.

These concepts for handling uncertainty are highly compatible with real-world applications[47, 208, 235, 263, 270, 278, 322]. For instance, neutrosophic sets extend fuzzy sets by introducing three membership degrees: truth, indeterminacy, and falsity, making them particularly valuable in scenarios with incomplete or conflicting information. Applications include:

- *Healthcare Decision-Making:* Neutrosophic sets assist in evaluating treatment options by balancing effectiveness (truth), uncertainty (indeterminacy), and risk (falsity) when data is incomplete or contradictory [29, 196].
- *Social Network Analysis:* They model relationships between users, such as trust, suspicion, and disagreement, in social networks [108, 253, 309, 382].
- *Fault Diagnosis in Engineering:* Neutrosophic sets identify faults in mechanical systems by accounting for uncertain and conflicting diagnostic evidence (cf.[155, 226, 326]).
- *Market Analysis:* Businesses use them to analyze customer preferences, integrating positive feedback (truth), ambiguous responses (indeterminacy), and negative feedback (falsity) [43, 264, 312].

This paper examines various models of uncertain graphs, including Fuzzy, Intuitionistic Fuzzy, Neutrosophic, and Plithogenic Graphs. These models extend classical graph theory by incorporating degrees of uncertainty, enabling a more nuanced analysis of ambiguous and complex relationships [120, 121, 123–127, 129, 131, 132].

Examples of uncertain graph models include the following:

- *Fuzzy Graph:* A Fuzzy Graph utilizes membership functions to represent uncertainty in vertices and edges, enabling more flexible modeling of relationships [8, 10, 12, 274, 306].
- *Neutrosophic Graph:* A Neutrosophic Graph extends Fuzzy Graphs by incorporating truth, indeterminacy, and falsity degrees for vertices and edges, offering a richer data representation [26, 63, 192, 272, 371, 372, 420]. It is well known that Neutrosophic Graphs can generalize Fuzzy Graphs.

- *Plithogenic Graph:* The Plithogenic Graph framework models graphs with multi-valued attributes using membership and contradiction functions, providing a detailed representation of complex relationships [121, 338, 357]. It is widely recognized that Plithogenic Graphs can generalize Neutrosophic Graphs.

These concepts, including set-based approaches, are applied in decision-making [18] as well as in neural networks [24, 112, 113, 416, 442] and machine learning[96, 142, 238, 246]. This highlights the importance of studying concepts related to uncertain graphs.

For reference, the relationships between Uncertain graphs are illustrated in Figure 2 (cf. [126]). Since Figure 2 is a highly simplified diagram, readers are encouraged to refer to the literature, such as [126], for further details if necessary.

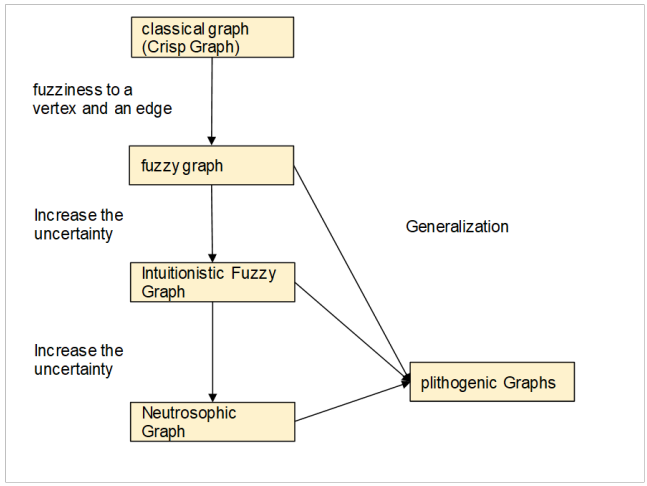

Fig. 2. Some Uncertain graphs Hierarchy(cf.[126]).

1.4 Our Contribution

This subsection highlights the key contributions of our work. While Graph Neural Networks (GNNs) for hypergraphs have been extensively studied, no previous research has explored the development of GNNs tailored to SuperHyperGraphs.

In this paper, we introduce the SuperHyperGraph Neural Network (SHGNN), a mathematical extension of Hypergraph Neural Networks that leverages the unique structural properties of SuperHyperGraphs. Additionally, we examine uncertain graph neural models, such as Neutrosophic Graph Neural Networks and Plithogenic Graph Neural Networks, which address similar challenges. Importantly, we demonstrate that both Neutrosophic and Plithogenic Graph Neural Networks serve as mathematical generalizations of Fuzzy Graph Neural Networks.

This work is theoretical in nature, focusing on establishing the mathematical framework for SHGNNs and PGNNs. It does not include computational experiments or practical implementations. Therefore, we hope that computational experiments will be conducted in the future by experts and readers alike. For precise definitions and detailed notations, readers are encouraged to consult the relevant literature, such as [115].

In this paper, we conduct a theoretical examination of the relationships between Graph Neural Networks, as illustrated in Figure 3. This diagram illustrates that the concept at the arrow's origin is included in (and generalized by) the concept at the arrow's destination.

Although not directly related to the Graph Neural Networks discussed earlier, this paper also explores several extended concepts in hypergraph theory, including Multilevel k-way Hypergraph Partitioning, Superhypergraph Random Walk, and the Superhypergraph Turán Problem. As these investigations are limited to theoretical considerations, it is hoped that computational experiments and practical validations will be conducted in the future as needed.

2 Preliminaries and Definitions

In this section, we provide a brief overview of the definitions and notations used throughout this paper. While we aim to make the content accessible to readers from various backgrounds, it is not possible to cover all relevant details comprehensively. Readers are encouraged to consult the referenced literature for additional information as needed.

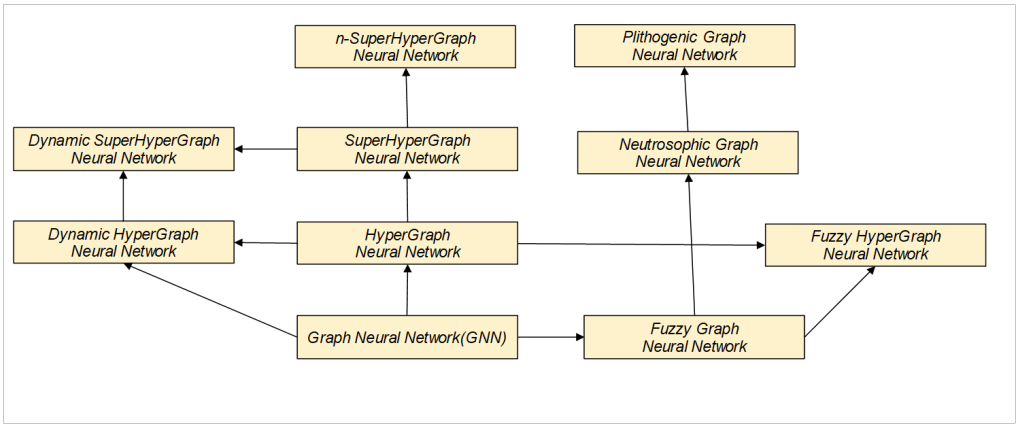

Fig. 3. Hierarchy of Some Neural Networks. This diagram illustrates that the concept at the arrow's origin is included in (and generalized by) the concept at the arrow's destination.

2.1 Basic Graph Concepts

This subsection outlines foundational graph concepts. For a comprehensive understanding of graph theory and notations, refer to [100–102, 158, 406]. Additionally, when discussing graph theory, basic set theory concepts are often used. Readers are encouraged to consult references such as [117, 182, 201, 389] as needed.

Definition 2.1 (Graph)**.** [102] A *graph* G is a mathematical structure defined as an ordered pair $G = (V, E)$, where:

- $V(G)$: the set of vertices (or nodes),
- $E(G)$: the set of edges, which represent connections between pairs of vertices.

Definition 2.2 (Degree)**.** [102] Let $G = (V, E)$ be a graph. The *degree* of a vertex $v \in V$, denoted $\deg(v)$, is the number of edges incident to v. For undirected graphs:

$$\deg(v) = |\{e \in E \mid v \in e\}|.$$

In directed graphs:

- The *in-degree* $\deg^-(v)$ is the number of edges directed into v.
- The *out-degree* $\deg^+(v)$ is the number of edges directed out of v.

Definition 2.3 (Subgraph)**.** [102] A *subgraph* G' of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that:

- $V' \subseteq V$,
- $E' \subseteq E \cap \{\{u, v\} \mid u, v \in V'\}$.

Definition 2.4 (Self-loop in an Undirected Graph)**.** In an undirected graph $G = (V, E)$, a *self-loop* is an edge that connects a vertex to itself. Formally, an edge $e \in E$ is a self-loop if $e = \{v, v\}$ for some $v \in V$.

Definition 2.5 (Real numbers)**.** (cf.[107, 303, 367]) The set of real numbers, denoted by $\mathbb{R}$, is defined as the unique complete ordered field. It satisfies the following:

- *Field Axioms:* $\mathbb{R}$ forms a field under addition and multiplication.
- *Order Axioms:* $\mathbb{R}$ is totally ordered and compatible with field operations.
- *Completeness Axiom:* Every non-empty subset of $\mathbb{R}$ that is bounded above has a least upper bound (supremum).

Definition 2.6 (Undirected Weighted Graph)**.** (cf.[66, 87, 259]) An *undirected weighted graph* $G = (V, E, w)$ is a graph where:

- V is the set of vertices.
- $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ is the set of undirected edges.
- $w : E \to \mathbb{R}^+$ is a weight function that assigns a non-negative weight to each edge $e \in E$.

Each edge $\{u, v\} \in E$ represents a bidirectional connection between u and v, and the weight $w(\{u, v\})$ indicates the strength, cost, or capacity of the connection.

2.2 Basic Definitions of Algorithm Complexity

This subsection introduces fundamental definitions for analyzing the algorithms described in later sections.

Definition 2.7 (Algorithms)**.** [320] Algorithms are step-by-step, well-defined procedures or rules for solving a problem or performing a task, often implemented in computing.

Definition 2.8 (Time Complexity)**.** (cf.[283, 320]) The *time complexity* of an algorithm is the total amount of computational time required to execute it, expressed as a function of the input size. Let $T(n, m)$ denote the time complexity for inputs of size n and m. The total time complexity is defined as:

$$T(n, m) = \max\{T_{\text{step1}}(n, m), T_{\text{step2}}(n, m), \ldots, T_{\text{stepk}}(n, m)\},$$

where $T_{\text{stepi}}(n, m)$ represents the time complexity of the i-th step of the algorithm.

Definition 2.9 (Space Complexity)**.** (cf.[283, 320]) The *space complexity* of an algorithm is the total amount of memory it requires, expressed as a function of the input size. This includes:

- *Input space*: memory required for storing the input data,
- *Auxiliary space*: additional memory for temporary variables and data structures used during computation.

Formally, the space complexity $S(n, m)$ is:

$$S(n, m) = S_{\text{input}}(n, m) + S_{\text{auxiliary}}(n, m).$$

Definition 2.10 (Big-O Notation)**.** (cf.[283, 320]) Big-O notation provides an asymptotic upper bound on the growth rate of a function. Let $f(n)$ and $g(n)$ be functions that map non-negative integers to non-negative real numbers. We write:

$$f(n) \in O(g(n))$$

if there exist positive constants $c > 0$ and $n_0 \geq 0$ such that:

$$f(n) \leq c \cdot g(n), \quad \forall n \geq n_0.$$

Readers may refer to the Lecture Notes or the Introduction for additional details as needed (cf.[1, 33, 86, 110, 173, 283, 320]).

2.3 Basic Graph Neural Network Concepts

Here are several definitions of Graph Neural Networks (GNNs). Readers may refer to the lecture notes or the introduction for further details(cf.[3, 94, 111, 205, 269, 297, 316, 324, 415, 440]).

Definition 2.11. (cf.[32,135,260]) A *matrix* is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns. Formally, an $m \times n$ matrix A is defined as:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

where:

- m is the number of rows,
- n is the number of columns,
- a_{ij} represents the element in the i-th row and j-th column.

Definition 2.12 (Adjacency Matrix)**.** (cf.[245, 414, 451]) The adjacency matrix of a graph $G = (V, E)$ with vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and edge set E is an $n \times n$ matrix $A = [a_{ij}]$, defined as:

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 2.13 (Weight matrix)**.** (cf.[276, 370]) A *weight matrix* is a matrix used in mathematical and computational models, particularly in neural networks, to represent the connection strengths between elements, such as nodes in a graph or neurons in a layer.

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the input data matrix, where:

- n is the number of data points (rows),
- d is the number of features (columns).

The weight matrix $\mathbf{W} \in \mathbb{R}^{d \times p}$ maps the input space to an output space, where:

- d is the dimension of the input features,
- p is the dimension of the output space.

The transformation is expressed as:

$$\mathbf{Z} = \mathbf{X}\mathbf{W},$$

where $\mathbf{Z} \in \mathbb{R}^{n \times p}$ is the resulting matrix in the output space.

In the context of neural networks or graph models, the entries w_{ij} in $\mathbf{W}$ represent the weight or strength of influence between the i-th input feature and the j-th output feature.

Definition 2.14 (Feature Vector)**.** (cf.[50,233,387]) Let O be an object or observation, and let $F = \{f_1, f_2, \ldots, f_n\}$ be a set of features, where $f_i : O \to \mathbb{R}$ is a function mapping O to the real numbers $\mathbb{R}$. A *feature vector* of O is defined as:

$$\mathbf{x} = [f_1(O), f_2(O), \ldots, f_n(O)]^\top \in \mathbb{R}^n,$$

where n is the number of features, and $\mathbf{x}$ is an element of the n-dimensional real vector space $\mathbb{R}^n$.

Definition 2.15 (Dataset)**.** (cf.[378]) A *dataset* is a finite set of data points. Formally, it is defined as:

$$D = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathcal{X}, i = 1, 2, \ldots, n\},$$

where $\mathbf{x}_i$ is the i-th data point in the input space $\mathcal{X}$, and n is the total number of data points.

Definition 2.16 (Normalization)**.** (cf.[36, 72, 109, 262, 384]) Normalization is a process of scaling a set of values to fit within a specific range, typically $[0, 1]$ or $[-1, 1]$. Given a dataset $\{x_1, x_2, \ldots, x_n\}$, normalization transforms each value x_i into a normalized value x_i' using the formula:

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)},$$

where:

- $\min(x) = \min\{x_1, x_2, \ldots, x_n\}$ is the minimum value in the dataset,
- $\max(x) = \max\{x_1, x_2, \ldots, x_n\}$ is the maximum value in the dataset.

If the range is $[-1, 1]$, the transformation is adjusted as:

$$x_i' = 2 \cdot \frac{x_i - \min(x)}{\max(x) - \min(x)} - 1.$$

Definition 2.17 (Graph Neural Network (GNN))**.** (cf.[449, 453]) Let $G = (V, E)$ be a graph, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges. Each vertex $v_i \in V$ is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$, and each edge $(v_i, v_j) \in E$ may optionally have a feature $\mathbf{e}_{ij} \in \mathbb{R}^k$.

A Graph Neural Network (GNN) computes node representations $\mathbf{h}_i^{(t)} \in \mathbb{R}^d$ at each layer t, using the graph structure and associated features.

Definition 2.18 (Key Components of Graph Neural Network)**.** (cf.[449,453]) Several key components of Graph Neural Networks are outlined below.

1. Node Initialization: At the initial layer ($t = 0$), the node representations are initialized as:

$$\mathbf{h}_i^{(0)} = \mathbf{x}_i, \quad \forall v_i \in V.$$

2. Message Passing(cf.[48, 228]): At each layer t, messages are exchanged between connected nodes. The messages received by a node v_i from its neighbors are computed as:

$$\mathbf{m}_i^{(t+1)} = \sum_{v_j \in \mathcal{N}(i)} \phi_m(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}),$$

where:

- $\mathcal{N}(i)$ is the set of neighbors of v_i,
- $\phi_m : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$ is the message function.

3. Node Update: (cf.[206]) The representation of each node is updated using the received messages:

$$\mathbf{h}_i^{(t+1)} = \phi_u(\mathbf{h}_i^{(t)}, \mathbf{m}_i^{(t+1)}),$$

where $\phi_u : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is the update function.

4. Readout Function: For graph-level tasks, a global representation $\mathbf{z}_G$ is computed by aggregating node representations:

$$\mathbf{z}_G = \phi_r\left(\{\mathbf{h}_i^{(T)} \mid v_i \in V\}\right),$$

where ϕ_r is the readout function (e.g., summation, averaging, or max-pooling).

Example 2.19 (Readout Function Examples)**.** (cf.[23,55,428]) A *readout function* ϕ_r computes a global representation of a graph by aggregating node representations. Below are some commonly used examples:

Mean Readout Function: (cf.[307,448]) The mean readout function computes the average of all node representations:

$$\phi_r\left(\{\mathbf{h}_i^{(T)} \mid v_i \in V\}\right) = \frac{1}{|V|} \sum_{v_i \in V} \mathbf{h}_i^{(T)},$$

where $\mathbf{h}_i^{(T)}$ is the final representation of node v_i at the last layer T.

Max-Pooling Readout Function: (cf.[27,301,452]) The max-pooling readout function selects the maximum value for each feature across all node representations:

$$\phi_r\left(\{\mathbf{h}_i^{(T)} \mid v_i \in V\}\right) = \max_{v_i \in V} \mathbf{h}_i^{(T)},$$

where the max operator is applied element-wise to the feature vectors.

Sum Readout Function: (cf.[89,308]) The sum readout function aggregates all node representations by summation:

$$\phi_r\left(\{\mathbf{h}_i^{(T)} \mid v_i \in V\}\right) = \sum_{v_i \in V} \mathbf{h}_i^{(T)}.$$

This function is particularly useful when the graph size varies, as it preserves the total magnitude of features.

Definition 2.20 (General Framework)**.** (cf.[449, 453]) The node update rule for all nodes at layer t can be expressed in matrix form:

$$\mathbf{H}^{(t+1)} = \phi_u\left(\mathbf{H}^{(t)}, \mathbf{A}, \mathbf{W}^{(t)}\right),$$

where:

- $\mathbf{H}^{(t)} \in \mathbb{R}^{n \times d}$ is the matrix of node representations,

- $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix,
- $\mathbf{W}^{(t)}$ are learnable weight matrices.

Definition 2.21 (Graph Convolutional Network)**.** (cf.[54,80,446,449,453]) For a Graph Convolutional Network (GCN), the propagation rule is:

$$\mathbf{H}^{(t+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(t)}\mathbf{W}^{(t)}\right),$$

where:

- $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ is the normalized adjacency matrix,
- $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops,
- $\tilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$,
- σ is an activation function (e.g., ReLU).

To understand Graph Convolutional Networks intuitively, consider the following example.

Example 2.22 (Graph Convolutional Network)**.** Imagine a social network(cf.[319]) where each person (node) has an attribute such as their interest in a specific topic (e.g., sports, music, or technology). Edges between nodes represent relationships or friendships between people. Each person also has initial attributes (node features), such as a score representing their interest in these topics.

The goal of the GCN is to predict a person's overall interest profile by combining their own features with information from their friends (neighboring nodes).

At each layer of the GCN:

1. The node collects information from its neighbors. For example, a sports enthusiast might update their profile based on their friends who are also interested in sports.
2. This information is aggregated using the normalized adjacency matrix $\hat{\mathbf{A}}$, ensuring that contributions from neighbors are weighted appropriately.
3. The aggregated information is then transformed using a learnable weight matrix $\mathbf{W}^{(t)}$, and a non-linear activation function σ is applied to introduce complexity to the model.

By stacking multiple layers of this process, each node gains a more comprehensive understanding of its broader neighborhood in the graph. For instance, after two layers, a person's profile reflects not only their immediate friends' interests but also those of their friends' friends.

This process allows GCNs to effectively learn and propagate information over the graph structure, making them powerful tools for tasks like node classification, graph classification, and link prediction.

2.4 Hypergraph Concepts

A hypergraph extends the concept of a traditional graph by allowing edges, called *hyperedges*, to connect any number of vertices, rather than being restricted to pairs[51,140,152–154]. This flexibility makes hypergraphs highly effective for modeling complex relationships in various domains, such as computer science and biology [114, 148, 195, 294]. The formal definitions are provided below.

Definition 2.23 (Hypergraph)**.** [51, 60] A *hypergraph* is a pair $H = (V(H), E(H))$, where:

- $V(H)$ is a nonempty set of vertices.
- $E(H)$ is a set of subsets of $V(H)$, called *hyperedges*. Each hyperedge $e \in E(H)$ can contain one or more vertices.

In this paper, we restrict our discussion to finite hypergraphs.

Example 2.24 (Hypergraph)**.** Let $H = (V(H), E(H))$ be a hypergraph with:

$$V(H) = \{v_1, v_2, v_3, v_4\}, \quad E(H) = \{\{v_1, v_2\}, \{v_2, v_3, v_4\}, \{v_1\}\}.$$

Here:

- $V(H)$ is the set of vertices: v_1, v_2, v_3, v_4.

- $E(H)$ is the set of hyperedges: $\{v_1, v_2\}$, $\{v_2, v_3, v_4\}$, and $\{v_1\}$.

Proposition 2.25. *A* hypergraph *is a generalized concept of a graph.*

Proof. This is evident. □

Definition 2.26 (subhypergraph)**.** [60] For a hypergraph $H = (V(H), E(H))$ and a subset $X \subseteq V(H)$, the *subhypergraph induced by* X is defined as:

$$H[X] = \big(X, \{e \cap X \mid e \in E(H)\}\big).$$

Additionally, the hypergraph obtained by removing the vertices in X is denoted as:

$$H \setminus X := H[V(H) \setminus X].$$

For further details on hypergraph notation and foundational concepts, refer to [60, 90].

2.5 SuperHyperGraph

A SuperHyperGraph is an advanced structure extending hypergraphs by allowing vertices and edges to be sets. The definition is provided below [340, 341].

Definition 2.27 (SuperHyperGraph [126,340,341])**.** Let V_0 be a finite set of base vertices. A *SuperHyperGraph* is an ordered pair $H = (V, E)$, where:

- $V \subseteq P(V_0)$ is a finite set of *supervertices*, each being a subset of V_0. That is, each supervertex $v \in V$ satisfies $v \subseteq V_0$.
- $E \subseteq P(V)$ is the set of *superedges*, where each superedge $e \in E$ is a subset of V, connecting multiple supervertices.

Example 2.28 (SuperHyperGraph)**.** Let $V_0 = \{x_1, x_2, x_3\}$ be the base vertex set. Define the supervertices as:

$$V = \{\{x_1, x_2\}, \{x_3\}, \{x_1\}\}.$$

Let the superedges be:

$$E = \{\{\{x_1, x_2\}, \{x_3\}\}, \{\{x_1\}, \{x_3\}\}\}.$$

Here:

- V contains subsets of V_0: $\{x_1, x_2\}$, $\{x_3\}$, $\{x_1\}$.
- E contains relationships among these supervertices: $\{\{x_1, x_2\}, \{x_3\}\}$ and $\{\{x_1\}, \{x_3\}\}$.

This SuperHypergraph extends the concept of a hypergraph by allowing supervertices (subsets of the base vertex set) to participate in superedges.

Proposition 2.29. *A* superhypergraph *is a generalized concept of a hypergraph.*

Proof. This is evident. □

Proposition 2.30. *A* superhypergraph *is a generalized concept of a graph.*

Proof. This is evident. □

When expressed concretely, including hypergraphs, a superhypergraph can be represented as follows. In this way, hypergraphs can be described and generalized using superhypergraphs.

Definition 2.31 (Expanded Hypergraph of a SuperHyperGraph)**.** Given a SuperHyperGraph $H = (V, E)$, the *Expanded Hypergraph* $H' = (V_0, E')$ is defined as follows:

- The vertex set is V_0, the set of base vertices.

- For each superedge $e \in E$, define the corresponding hyperedge $e' \in E'$ by

$$e' = \bigcup_{v \in e} v,$$

where $v \in V$ are supervertices in e. Then

$$E' = \{e' \mid e \in E\}.$$

Example 2.32 (Expanded Hypergraph)**.** Consider the SuperHyperGraph $H = (V, E)$ defined as follows:

- The base vertex set is $V_0 = \{x_1, x_2, x_3\}$.
- The supervertices are:
$$V = \{\{x_1, x_2\}, \{x_3\}, \{x_1\}\}.$$
- The superedges are:
$$E = \{\{\{x_1, x_2\}, \{x_3\}\}, \{\{x_1\}, \{x_3\}\}\}.$$

The Expanded Hypergraph $H' = (V_0, E')$ is constructed as follows:

- The vertex set remains $V_0 = \{x_1, x_2, x_3\}$, which is the base vertex set.
- For each superedge $e \in E$, the corresponding hyperedge e' is obtained by taking the union of all supervertices v in e:
$$e' = \bigcup_{v \in e} v.$$
- The expanded edge set E' is:

$$e'_1 = \bigcup_{v \in \{\{x_1, x_2\}, \{x_3\}\}} v = \{x_1, x_2\} \cup \{x_3\} = \{x_1, x_2, x_3\},$$
$$e'_2 = \bigcup_{v \in \{\{x_1\}, \{x_3\}\}} v = \{x_1\} \cup \{x_3\} = \{x_1, x_3\}.$$

Thus, the expanded edge set is:
$$E' = \{\{x_1, x_2, x_3\}, \{x_1, x_3\}\}.$$

To summarize:

- The Expanded Hypergraph H' has the vertex set:
$$V_0 = \{x_1, x_2, x_3\}.$$
- The edge set is:
$$E' = \{\{x_1, x_2, x_3\}, \{x_1, x_3\}\}.$$

This construction illustrates how the supervertices and superedges in a SuperHyperGraph are transformed into vertices and edges in the corresponding Expanded Hypergraph.

Theorem 2.33. *The Expanded Hypergraph of a SuperHyperGraph generalizes a Hypergraph.*

Proof. Let $H = (V, E)$ be a SuperHyperGraph with V as the set of supervertices, where each supervertex $v \in V$ is a subset of a base vertex set V_0. Let $H' = (V_0, E')$ be the Expanded Hypergraph derived from H, where:

$$E' = \{e' \mid e' = \bigcup_{v \in e} v,\ e \in E\}.$$

To prove that the Expanded Hypergraph H' generalizes a Hypergraph, consider the following cases:

Case 1: SuperHyperGraph reduces to a Hypergraph. If each supervertex $v \in V$ corresponds to exactly one base vertex in V_0, then $V = V_0$. In this case, each superedge $e \in E$ is a subset of V_0, and the expansion rule:

$$e' = \bigcup_{v \in e} v$$

yields $e' = e$. Therefore, $H' = (V_0, E')$ is identical to the original Hypergraph H, showing that the Expanded Hypergraph is equivalent to a Hypergraph when H is already a Hypergraph.

Case 2: General SuperHyperGraph. When H is a general SuperHyperGraph, each supervertex $v \in V$ may represent a subset of V_0. The expansion process aggregates all base vertices in V_0 that are part of the supervertices in each superedge $e \in E$. This allows $H' = (V_0, E')$ to represent relationships among base vertices in V_0 in a way that subsumes the structure of a Hypergraph.

The Expanded Hypergraph H' retains the flexibility to represent any Hypergraph by treating each vertex $v \in V$ as a single base vertex in V_0. Simultaneously, it extends the concept of a Hypergraph by allowing vertices in E to represent subsets of base vertices, enabling more complex relational structures.

Since the Expanded Hypergraph H' encompasses both the structure of Hypergraphs and the extended relational complexity of SuperHyperGraphs, we conclude that the Expanded Hypergraph of a SuperHyperGraph generalizes a Hypergraph. □

2.6 HGNN:Hypergraph Neural Network

The Hypergraph Neural Network is a concept designed to utilize the general Graph Neural Network at a higher level, and it has been studied extensively across numerous frameworks and concepts[115, 229, 231, 236, 239, 239, 244, 410, 425, 426]. The definitions are provided below.

Definition 2.34 (Hypergraph Neural Network)**.** [115] Let $G = (V, E, W)$ be a hypergraph, where:

- $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices.
- $E = \{e_1, e_2, \ldots, e_m\}$ is the set of hyperedges, where each hyperedge $e_i \subseteq V$ connects a subset of vertices.
- $W = \text{diag}(w_1, w_2, \ldots, w_m)$ is a diagonal matrix of hyperedge weights, where $w_i > 0$ represents the weight of hyperedge e_i.

The *Hypergraph Neural Network (HGNN)* is a neural network framework designed for representation learning on hypergraphs. It utilizes the hypergraph structure to aggregate features from vertices and their connections through hyperedges. The key components of HGNN are defined as follows:

Incidence Matrix The incidence matrix $H \in \mathbb{R}^{n \times m}$ of the hypergraph G is defined as:

$$H_{ij} = \begin{cases} 1, & \text{if vertex } v_i \in e_j, \\ 0, & \text{otherwise.} \end{cases}$$

Vertex and Hyperedge Degrees The degree of a vertex $v_i \in V$ is defined as:

$$d(v_i) = \sum_{e_j \in E} H_{ij}\, w_j.$$

The degree of a hyperedge $e_j \in E$ is defined as:

$$\delta(e_j) = \sum_{v_i \in V} H_{ij}.$$

Let $D_V \in \mathbb{R}^{n \times n}$ and $D_E \in \mathbb{R}^{m \times m}$ be the diagonal matrices of vertex degrees and hyperedge degrees, respectively, where:

$$(D_V)_{ii} = d(v_i), \quad (D_E)_{jj} = \delta(e_j).$$

Hypergraph Laplacian (cf.[75, 137]) The hypergraph Laplacian Δ is defined as:

$$\Delta = I - D_V^{-1/2} H W D_E^{-1} H^\top D_V^{-1/2},$$

where I is the identity matrix.

Spectral Convolution on Hypergraph (cf.[38, 251]) The convolution operation in HGNN is performed in the spectral domain using the hypergraph Laplacian. Given a feature matrix $X \in \mathbb{R}^{n \times d}$, where each row x_i represents the feature vector of vertex v_i, the output feature matrix $Y \in \mathbb{R}^{n \times c}$ is computed as:

$$Y = \sigma\left(D_V^{-1/2} H W D_E^{-1} H^\top D_V^{-1/2} X \Theta\right),$$

where:

- σ is a nonlinear activation function (e.g., ReLU).
- $\Theta \in \mathbb{R}^{d \times c}$ is the learnable weight matrix.

Node Classification Task For a node classification task, let $X^{(0)}$ be the input feature matrix. A multi-layer HGNN can be defined recursively as:

$$X^{(l+1)} = \sigma\left(D_V^{-1/2} H W D_E^{-1} H^\top D_V^{-1/2} X^{(l)} \Theta^{(l)}\right),$$

where l denotes the layer index, $\Theta^{(l)}$ is the learnable weight matrix for layer l, and $X^{(l+1)}$ is the feature matrix output at layer $l + 1$.

Output Layer In the final layer, the softmax function is applied to the output features to produce class probabilities for each node:

$$\hat{Y} = \text{softmax}(X^{(L)}),$$

where L is the total number of layers and $\hat{Y} \in \mathbb{R}^{n \times c}$ contains the predicted probabilities for c classes.

Proposition 2.35. *A Hypergraph Neural Network can generalize a Classical Graph Neural Network.*

Proof. This is evident from the definitions. □

2.7 Uncertain Graph

The concept of the Fuzzy Set, introduced approximately half a century ago, has spurred the development of various graph theories aimed at modeling uncertainty[430]. In this section, we outline definitions for several frameworks, including Fuzzy Graphs, Intuitionistic Fuzzy Graphs, Neutrosophic Graphs, and Single-Valued Pentapartitioned Neutrosophic Graphs.

A Fuzzy Graph is frequently analyzed in the context of a Crisp Graph [121]. To provide a foundation, we begin by presenting the definition of a Crisp Graph [121].

Definition 2.36 (Crisp Graph)**.** (cf.[121]) A *Crisp Graph* $G = (V, E)$ is defined as follows:

1. V: A non-empty finite set of vertices (or nodes).
2. $E \subseteq \{\{u, v\} \mid u, v \in V \text{ and } u \neq v\}$: A set of unordered pairs of vertices, called edges. Each edge is associated with exactly two vertices, referred to as its endpoints. An edge is said to connect its endpoints.

Special Cases

- A graph G with $E = \emptyset$ is called an *edgeless graph*.

Next, we introduce the concepts of Fuzzy Graph, Intuitionistic Fuzzy Graph, Neutrosophic Graph, Hesitant Fuzzy Graph, Quadripartitioned Neutrosophic Graph (QNG), and Single-Valued Pentapartitioned Neutrosophic Graph. Readers are encouraged to refer to survey papers (e.g., [121, 123]) for more detailed information if needed.

Definition 2.37 (Unified Framework for Uncertain Graphs)**.** (cf. [123]) Let $G = (V, E)$ be a classical graph, where V is the set of vertices and E is the set of edges. Depending on the type of graph, each vertex $v \in V$ and edge $e \in E$ is associated with membership values to represent various degrees of truth, indeterminacy, falsity, and other measures of uncertainty.

1. *Fuzzy Graph* (cf. [53, 136, 144, 267, 279, 306, 404])
    - Each vertex $v \in V$ is assigned a membership degree $\sigma(v) \in [0, 1]$.

- Each edge $e = (u, v) \in E$ is assigned a membership degree $\mu(u, v) \in [0, 1]$.

2. *Intuitionistic Fuzzy Graph (IFG)* (cf. [9, 199, 383, 445])
   - Each vertex $v \in V$ has two values: $\mu_A(v) \in [0, 1]$ (degree of membership) and $\nu_A(v) \in [0, 1]$ (degree of non-membership), satisfying $\mu_A(v) + \nu_A(v) \leq 1$.
   - Each edge $e = (u, v) \in E$ has two values: $\mu_B(u, v) \in [0, 1]$ and $\nu_B(u, v) \in [0, 1]$, with $\mu_B(u, v) + \nu_B(u, v) \leq 1$.
3. *Neutrosophic Graph* (cf. [17, 65, 161, 188, 209, 341, 354])
   - Each vertex $v \in V$ is associated with a triplet
   $$\sigma(v) = (\sigma_T(v), \sigma_I(v), \sigma_F(v))$$
   , where
   $$\sigma_T(v), \sigma_I(v), \sigma_F(v) \in [0, 1]$$
   and $\sigma_T(v) + \sigma_I(v) + \sigma_F(v) \leq 3$.
   - Each edge $e = (u, v) \in E$ is associated with a triplet $\mu(e) = (\mu_T(e), \mu_I(e), \mu_F(e))$.
4. *Hesitant Fuzzy Graph* (cf. [39, 146, 281, 286, 417])
   - Each vertex $v \in V$ is assigned a hesitant fuzzy set $\sigma(v) \subseteq [0, 1]$.
   - Each edge $e = (u, v) \in E$ is assigned a hesitant fuzzy set $\mu(e) \subseteq [0, 1]$.
5. *Quadripartitioned Neutrosophic Graph (QNG)* (cf. [190, 191, 193, 313, 327])
   - Each vertex $v \in V$ is associated with a quadripartitioned neutrosophic membership
   $$\sigma(v) = (\sigma_1(v), \sigma_2(v), \sigma_3(v), \sigma_4(v))$$
   , where
   $$\sigma_1(v), \sigma_2(v), \sigma_3(v), \sigma_4(v) \in [0, 1]$$
   and
   $$\sigma_1(v) + \sigma_2(v) + \sigma_3(v) + \sigma_4(v) \leq 4$$
   .
   - Each edge $e = (u, v) \in E$ is associated with a quadripartitioned membership
   $$\sigma(e) = (\sigma_1(e), \sigma_2(e), \sigma_3(e), \sigma_4(e))$$
   , satisfying:
   $$\begin{aligned} \sigma_1(e) &\leq \min\{\sigma_1(u), \sigma_1(v)\}, \\ \sigma_2(e) &\leq \min\{\sigma_2(u), \sigma_2(v)\}, \\ \sigma_3(e) &\leq \max\{\sigma_3(u), \sigma_3(v)\}, \\ \sigma_4(e) &\leq \max\{\sigma_4(u), \sigma_4(v)\}. \end{aligned}$$
6. *Single-Valued Pentapartitioned Neutrosophic Graph* (cf. [91, 189, 191, 298])
   - Each vertex $v \in V$ is assigned a quintuple
   $$\sigma(v) = (\sigma_1(v), \sigma_2(v), \sigma_3(v), \sigma_4(v), \sigma_5(v))$$
   , where
   $$\sigma_1(v), \sigma_2(v), \sigma_3(v), \sigma_4(v), \sigma_5(v) \in [0, 1]$$
   and
   $$\sigma_1(v) + \sigma_2(v) + \sigma_3(v) + \sigma_4(v) + \sigma_5(v) \leq 5$$
   .

- Each edge $e = (u, v) \in E$ is assigned a quintuple

$$\sigma(e) = (\sigma_1(e), \sigma_2(e), \sigma_3(e), \sigma_4(e), \sigma_5(e))$$

, satisfying:

$$\begin{aligned}
\sigma_1(e) &\leq \min\{\sigma_1(u), \sigma_1(v)\}, \\
\sigma_2(e) &\leq \min\{\sigma_2(u), \sigma_2(v)\}, \\
\sigma_3(e) &\geq \max\{\sigma_3(u), \sigma_3(v)\}, \\
\sigma_4(e) &\geq \max\{\sigma_4(u), \sigma_4(v)\}, \\
\sigma_5(e) &\geq \max\{\sigma_5(u), \sigma_5(v)\}.
\end{aligned}$$

We provide examples of Fuzzy Graphs and Neutrosophic Graphs applied to real-world scenarios. These examples demonstrate how Uncertain Graphs are well-known for their ability to model various phenomena in the real world[7, 18, 64, 160, 192, 329].

Example 2.38 (Fuzzy Graph: Social Network with Varying Friendship Strengths)**.** Consider a social network where individuals are connected based on their friendships, with varying strengths (cf.[248, 252, 310, 402]). This can be modeled using a fuzzy graph, where vertices represent individuals, and edges represent friendships with varying degrees of strength.

Definition: Let $G = (V, E)$ be a fuzzy graph where:

- $V = \{\text{Alice}, \text{Bob}, \text{Carol}, \text{Dave}\}$ is the set of individuals.
- $E \subseteq V \times V$ represents the friendships between individuals.

Membership Functions:

- *Vertex Membership Degrees ($\sigma(v)$):* The membership degree of each vertex represents the individual's level of activity or influence in the social network:

$$\begin{aligned}
\sigma(\text{Alice}) &= 0.9 \quad \text{(Highly active user)}, \\
\sigma(\text{Bob}) &= 0.7 \quad \text{(Active user)}, \\
\sigma(\text{Carol}) &= 0.5 \quad \text{(Moderately active user)}, \\
\sigma(\text{Dave}) &= 0.3 \quad \text{(Less active user)}.
\end{aligned}$$

- *Edge Membership Degrees ($\mu(u, v)$):* The membership degree of each edge represents the strength of the friendship:

$$\begin{aligned}
\mu(\text{Alice}, \text{Bob}) &= 0.8 \quad \text{(Strong friendship)}, \\
\mu(\text{Bob}, \text{Carol}) &= 0.6 \quad \text{(Moderate friendship)}, \\
\mu(\text{Carol}, \text{Dave}) &= 0.4 \quad \text{(Weak friendship)}, \\
\mu(\text{Alice}, \text{Dave}) &= 0.2 \quad \text{(Very weak friendship)}.
\end{aligned}$$

Alice is highly active in the network, engaging frequently, while Dave is the least active. Alice and Bob share a strong friendship, while Carol and Dave have a weak connection.

This fuzzy graph allows for a nuanced analysis of social networks by modeling the varying strengths of relationships and activity levels, aiding in tasks like community detection or recommendation systems (cf.[71, 93, 409, 413]).

Example 2.39 (Neutrosophic Graph: Disease Transmission Network with Uncertainty)**.** In epidemiology, understanding the spread of disease through a population is crucial. A neutrosophic graph can model the uncertainty in infection statuses and transmission probabilities (cf.[4, 270, 328]).

Definition: Let $G = (V, E)$ be a neutrosophic graph where:

- $V = \{\text{Patient1}, \text{Patient2}, \text{Patient3}, \text{Patient4}\}$ represents individuals.
- $E \subseteq V \times V$ represents potential transmission paths.

Membership Functions:

- *Vertex Membership Triplets ($\sigma(v) = (\sigma_T(v), \sigma_I(v), \sigma_F(v))$):* Each vertex is assigned degrees of truth (σ_T), indeterminacy (σ_I), and falsity (σ_F):

$$\begin{aligned}
&\sigma(\text{Patient1}) = (0.9, 0.1, 0.0) \quad \text{(Highly likely infected)},\\
&\sigma(\text{Patient2}) = (0.5, 0.4, 0.1) \quad \text{(Uncertain status)},\\
&\sigma(\text{Patient3}) = (0.2, 0.3, 0.5) \quad \text{(Possibly not infected)},\\
&\sigma(\text{Patient4}) = (0.0, 0.1, 0.9) \quad \text{(Highly likely not infected)}.
\end{aligned}$$

- *Edge Membership Triplets ($\mu(e) = (\mu_T(e), \mu_I(e), \mu_F(e))$):* Each edge is assigned degrees of truth, indeterminacy, and falsity:

$$\begin{aligned}
&\mu(\text{Patient1}, \text{Patient2}) = (0.8, 0.1, 0.1) \quad \text{(High likelihood of transmission)},\\
&\mu(\text{Patient2}, \text{Patient3}) = (0.4, 0.4, 0.2) \quad \text{(Uncertain transmission)},\\
&\mu(\text{Patient3}, \text{Patient4}) = (0.1, 0.2, 0.7) \quad \text{(Low likelihood of transmission)},\\
&\mu(\text{Patient1}, \text{Patient4}) = (0.2, 0.3, 0.5) \quad \text{(Possible but unlikely transmission)}.
\end{aligned}$$

Patient1 is highly likely infected and may transmit the disease to Patient2. The transmission between Patient2 and Patient3 is uncertain. Patient4 is highly unlikely to be infected, with low chances of transmission from others.

Neutrosophic graphs can aid in modeling uncertain infection and transmission dynamics, supporting efforts in contact tracing, resource allocation, and risk assessment.

Proposition 2.40. *Neutrosophic graphs can generalize Fuzzy Graphs.*

Proof. This follows directly (cf.[355]). □

A Plithogenic Graph is a generalized graph based on the concept of a Plithogenic Set. This graph is known for its ability to generalize structures such as Fuzzy Graphs and Neutrosophic Graphs described earlier. The definition is provided below [338].

Definition 2.41. [145, 338, 339, 357, 364] Let $G = (V, E)$ be a crisp graph where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. A *Plithogenic Graph PG* is defined as:

$$PG = (PM, PN)$$

where:

1. *Plithogenic Vertex Set* $PM = (M, l, Ml, adf, aCf)$:
   - $M \subseteq V$ is the set of vertices.
   - l is an attribute associated with the vertices.
   - Ml is the range of possible attribute values.
   - $adf : M \times Ml \to [0, 1]^s$ is the *Degree of Appurtenance Function (DAF)* for vertices.
   - $aCf : Ml \times Ml \to [0, 1]^t$ is the *Degree of Contradiction Function (DCF)* for vertices.
2. *Plithogenic Edge Set* $PN = (N, m, Nm, bdf, bCf)$:
   - $N \subseteq E$ is the set of edges.
   - m is an attribute associated with the edges.
   - Nm is the range of possible attribute values.
   - $bdf : N \times Nm \to [0, 1]^s$ is the *Degree of Appurtenance Function (DAF)* for edges.
   - $bCf : Nm \times Nm \to [0, 1]^t$ is the *Degree of Contradiction Function (DCF)* for edges.

The Plithogenic Graph PG must satisfy the following conditions:

1. *Edge Appurtenance Constraint*: For all $(x, a), (y, b) \in M \times Ml$:

$$bdf\left((xy), (a, b)\right) \leq \min\{adf(x, a), adf(y, b)\}$$

where $xy \in N$ is an edge between vertices x and y, and $(a, b) \in Nm \times Nm$ are the corresponding attribute values.

2. *Contradiction Function Constraint*: For all $(a, b), (c, d) \in Nm \times Nm$:

$$bCf\left((a, b), (c, d)\right) \leq \min\{aCf(a, c), aCf(b, d)\}$$

3. *Reflexivity and Symmetry of Contradiction Functions*:

$$\begin{aligned} aCf(a, a) &= 0, && \forall a \in Ml \\ aCf(a, b) &= aCf(b, a), && \forall a, b \in Ml \\ bCf(a, a) &= 0, && \forall a \in Nm \\ bCf(a, b) &= bCf(b, a), && \forall a, b \in Nm \end{aligned}$$

Example 2.42. (cf.[121]) The following examples of Plithogenic Graphs are provided.

- When $s = t = 1$, PG is called a *Plithogenic Fuzzy Graphs*.
- When $s = 2, t = 1$, PG is called a *Plithogenic Intuitionistic Fuzzy Graphs*.
- When $s = 3, t = 1$, PG is called a *Plithogenic Neutrosophic Graphs*.
- When $s = 4, t = 1$, PG is called a *Plithogenic quadripartitioned Neutrosophic Graphs* (cf.[193, 302, 327]).
- When $s = 5, t = 1$, PG is called a *Plithogenic pentapartitioned Neutrosophic Graphs* (cf.[56, 92, 256]).
- When $s = 6, t = 1$, PG is called a *Plithogenic hexapartitioned Neutrosophic Graphs* (cf.[287]).
- When $s = 7, t = 1$, PG is called a *Plithogenic heptapartitioned Neutrosophic Graphs* (cf.[62, 271]).
- When $s = 8, t = 1$, PG is called a *Plithogenic octapartitioned Neutrosophic Graphs*.
- When $s = 9, t = 1$, PG is called a *Plithogenic nonapartitioned Neutrosophic Graphs*.

2.8 Fuzzy Graph Neural Network (F-GNN)

In this subsection, we introduce the concept of the Fuzzy Graph Neural Network (F-GNN). A Fuzzy Graph Neural Network (F-GNN) is a graph inference model that combines the principles of fuzzy logic and graph neural networks (GNNs). It is specifically designed to address fuzzy and uncertain data within graph-structured information (cf.[78, 116, 162, 224, 295, 392, 439, 442]). Below, we present the formal definition of F-GNN.

Definition 2.43. [104] An F-GNN is defined as a quintuple:

$$\text{F-GNN} = \left(G, \mathcal{F}_V, \mathcal{F}_E, \mathcal{R}, \mathcal{D}\right),$$

where:

- $G = (V, E)$ is a graph where V represents the set of vertices and E represents the set of edges.
- $\mathcal{F}_V$ and $\mathcal{F}_E$ are the fuzzification functions for vertices and edges, respectively. These functions map vertex and edge attributes to fuzzy membership values:

$$\mathcal{F}_V : \mathcal{X}_V \to [0, 1]^M, \quad \mathcal{F}_E : \mathcal{X}_E \to [0, 1]^M,$$

where M is the number of fuzzy subsets, and $\mathcal{X}_V$ and $\mathcal{X}_E$ denote the attribute spaces for vertices and edges.

- $\mathcal{R}$ represents the rule layer, which encodes fuzzy rules of the form:

$$\text{IF} \bigwedge_{i=1}^{N} \text{vertex } v_i \text{ satisfies } \mathcal{F}_V(v_i) \text{ THEN } \mathcal{D}(v_i) \text{ outputs the prediction,}$$

  where $\mathcal{D}$ is the defuzzification layer.

- $\mathcal{D}$ is the defuzzification function, which aggregates the outputs of the rule layer to produce a crisp output for each vertex or edge.

Definition 2.44. [104] Given an input graph $G = (V, E)$ with vertex features X_V and edge features X_E, F-GNN operates as follows:

1. *Fuzzification Layer:* Each vertex $v \in V$ and edge $e \in E$ is fuzzified using membership functions:

$$\mathcal{F}_V(v) = [\mu_1(v), \mu_2(v), \ldots, \mu_M(v)], \quad \mathcal{F}_E(e) = [\mu_1(e), \mu_2(e), \ldots, \mu_M(e)].$$

2. *Rule Layer:* A set of fuzzy rules is defined to aggregate neighborhood information. For example:

$$\text{IF } v \in A_m \text{ AND } u \in A_n \text{ THEN } y_k = f_k(x_v, x_u),$$

   where A_m, A_n are fuzzy subsets, x_v, x_u are vertex features, and f_k is a trainable function.

3. *Normalization Layer:* The firing strength of each rule is normalized:

$$\hat{r}_k = \frac{r_k}{\sum_{j=1}^{K} r_j},$$

   where r_k is the firing strength of the k-th rule.

4. *Defuzzification Layer:* The normalized rule outputs are aggregated to produce crisp predictions:

$$y = \sum_{k=1}^{K} \hat{r}_k \cdot f_k(x).$$

Definition 2.45. [104] For a multi-layer F-GNN, the l-th layer is defined as:

$$H^{(l)} = \sigma\left(f_\theta\left(H^{(l-1)}, A\right) + H^{(l-1)}\right),$$

where:

- $H^{(l)}$ is the output of the l-th layer.
- σ is a non-linear activation function (e.g., ReLU).
- A is the adjacency matrix of the graph.
- f_θ is a trainable function.

  The final output of the F-GNN is:

$$Y = \text{Softmax}\left(H^{(L)}\right),$$

where L is the number of layers in the F-GNN.

Theorem 2.46. *A Fuzzy Graph Neural Network (F-GNN) generalizes a Graph Neural Network (GNN).*

Proof. To prove this, we show that the definition of an F-GNN encompasses the definition of a GNN as a special case.

1. Graph Structure: Both GNNs and F-GNNs operate on a graph $G = (V, E)$, where V is the set of vertices, and $E \subseteq V \times V$ is the set of edges. While GNNs use crisp edge connections, F-GNNs extend this by assigning fuzzy membership values to vertices and edges through the fuzzification functions $\mathcal{F}_V$ and $\mathcal{F}_E$:

$$\mathcal{F}_V : \mathcal{X}_V \to [0,1]^M, \quad \mathcal{F}_E : \mathcal{X}_E \to [0,1]^M.$$

When $M = 1$ and membership values are restricted to binary $\{0, 1\}$, the F-GNN reduces to a standard GNN, where $\mathcal{F}_V$ and $\mathcal{F}_E$ represent crisp vertices and edges.

2. Message Passing: In a GNN, messages between nodes are exchanged using functions ϕ_m and aggregated at each node v_i as:

$$\mathbf{m}_i^{(t+1)} = \sum_{v_j \in \mathcal{N}(i)} \phi_m(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}),$$

where $\mathcal{N}(i)$ is the set of neighbors of v_i.

In an F-GNN, the message passing incorporates fuzzy membership values through the rule layer $\mathcal{R}$, which defines fuzzy rules such as:

$$\text{IF } v_i \in A_m \text{ AND } v_j \in A_n \text{ THEN } f_k(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij}),$$

where A_m and A_n are fuzzy subsets, and f_k is a trainable function. If fuzzy subsets A_m and A_n are crisp (e.g., $A_m = A_n = \{1\}$), the F-GNN reduces to the standard message passing mechanism of a GNN.

3. Node Updates: In a GNN, node updates are defined as:

$$\mathbf{h}_i^{(t+1)} = \phi_u(\mathbf{h}_i^{(t)}, \mathbf{m}_i^{(t+1)}),$$

where ϕ_u is a node update function.

In an F-GNN, node updates are governed by fuzzy rules and defuzzification, aggregating over normalized firing strengths:

$$y = \sum_{k=1}^{K} \hat{r}_k \cdot f_k(\mathbf{h}_i),$$

where $\hat{r}_k$ is the normalized firing strength of the k-th fuzzy rule. If there is only one rule ($K = 1$) and no fuzzification is applied, the F-GNN node update simplifies to the standard GNN node update.

4. Generalization: The fuzzification and defuzzification layers in an F-GNN extend the crisp operations of a GNN by introducing degrees of membership, enabling the model to handle uncertainty and imprecision. When these additional features are disabled (e.g., by setting $M = 1$ and $K = 1$), the F-GNN reduces exactly to a GNN.

Since every operation in a GNN is a special case of the corresponding operation in an F-GNN, we conclude that the F-GNN generalizes the GNN. □

3 Result: SuperHypergraph Neural Network

In this section, we explore the SuperHyperGraph Neural Network.

3.1 SuperHypergraph Neural Network

In this subsection, we explore the definition and theoretical framework of the SuperHypergraph Neural Network. This concept is a mathematical extension of the Hypergraph Neural Network. It is important to note that this study is purely theoretical, with no practical implementation or testing conducted on actual systems.

Definition 3.1 (SuperHypergraph Neural Network)**.** Let $H = (V, E)$ be a SuperHyperGraph with base vertices V_0, and let $H' = (V_0, E')$ be its Expanded Hypergraph. Let $X \in \mathbb{R}^{|V_0| \times d}$ be the feature matrix for the base vertices. Define:

- The incidence matrix $H' \in \mathbb{R}^{|V_0| \times |E'|}$ with entries

$$H'_{ij} = \begin{cases} 1, & \text{if } v_i \in e'_j, \\ 0, & \text{otherwise.} \end{cases}$$

- The diagonal vertex degree matrix $D_V \in \mathbb{R}^{|V_0|\times|V_0|}$ with entries

$$(D_V)_{ii} = d_V(v_i) = \sum_{j=1}^{|E'|} H'_{ij}\, w(e'_j),$$

  where $w(e'_j)$ is the weight of hyperedge e'_j.

- The diagonal hyperedge degree matrix $D_E \in \mathbb{R}^{|E'|\times|E'|}$ with entries

$$(D_E)_{jj} = d_E(e'_j) = \sum_{i=1}^{|V_0|} H'_{ij}.$$

The *convolution operation* in the SHGNN is defined as

$$Y = \sigma\left(D_V^{-1/2} H' W D_E^{-1} H'^{\top} D_V^{-1/2} X\Theta\right),$$

where:

- $Y \in \mathbb{R}^{|V_0|\times c}$ is the output feature matrix.
- $W \in \mathbb{R}^{|E'|\times|E'|}$ is the diagonal matrix of hyperedge weights.
- $\Theta \in \mathbb{R}^{d\times c}$ is the learnable weight matrix.
- σ is an activation function (e.g., ReLU[44, 234]).

Theorem 3.2. *A SuperHypergraph Neural Network (SHGNN) inherently possesses the structure of a SuperHyperGraph $H = (V, E)$, where:*

1. *The vertex set V corresponds to the subsets of the base vertices V_0 used in the SHGNN.*
2. *The edge set E corresponds to the relationships (superedges) among the supervertices, as encoded in the hyperedge-weighted incidence matrix H'.*

Proof. By definition, the SuperHyperGraph vertex set $V \subseteq P(V_0)$ consists of subsets of the base vertex set V_0. In the SHGNN, the input feature matrix $X \in \mathbb{R}^{|V_0|\times d}$ defines the features associated with each base vertex $v_i \in V_0$. These features are subsequently aggregated and processed in layers, preserving the subset structure of V.

The edge set E in a SuperHyperGraph is defined as $E \subseteq P(V)$, connecting multiple supervertices. In the SHGNN, the relationships between subsets (supervertices) are captured by the hyperedges $e \in E$, represented in the weighted incidence matrix H'. The matrix H' explicitly encodes whether a base vertex $v_i \in V_0$ belongs to a hyperedge $e'_j \in E'$, thereby maintaining the SuperHyperGraph's structure.

The convolution operation in the SHGNN, defined as:

$$Y = \sigma\left(D_V^{-1/2} H' W D_E^{-1} H'^{\top} D_V^{-1/2} X\Theta\right),$$

propagates and updates features across the graph while preserving the structural relationships encoded in H. This operation respects the adjacency relationships among subsets of V_0 as defined by the superedges.

The SHGNN's architecture, including its vertex and edge representations and layer-wise operations, directly corresponds to the mathematical structure of a SuperHyperGraph $H = (V, E)$. Therefore, the SHGNN inherently possesses the structure of a SuperHyperGraph. □

Theorem 3.3. *The Hypergraph Neural Network (HGNN) is a special case of the SuperHypergraph Neural Network (SHGNN). Specifically, when all supervertices are singleton subsets of V_0, and all superedges connect these singleton supervertices, the SHGNN reduces to the HGNN.*

Proof. Assume that all supervertices are singletons, i.e.,

$$V = \{\{v_i\} \mid v_i \in V_0\}.$$

Then, each superedge $e \in E$ connects supervertices that correspond directly to base vertices in V_0.

For each superedge $e \in E$, the corresponding hyperedge in the Expanded Hypergraph is

$$e' = \bigcup_{v \in e} v = \bigcup_{v \in e} \{v_i\} = \{v_i \mid v = \{v_i\} \in e\}.$$

Thus, the Expanded Hypergraph $H' = (V_0, E')$ is identical to the original hypergraph defined over V_0 with hyperedges E'.

The convolution operation in SHGNN becomes

$$Y = \sigma\left(D_V^{-1/2} H W D_E^{-1} H^\top D_V^{-1/2} X\Theta\right),$$

which is exactly the convolution operation used in the Hypergraph Neural Network (HGNN) .

Therefore, the SHGNN reduces to the HGNN in this case, demonstrating that SHGNN generalizes HGNN. □

Corollary 3.4. *The Graph Convolutional Network (GCN) is a special case of the SHGNN when all hyperedges connect exactly two vertices.*

Proof. When all hyperedges e'_j in the Expanded Hypergraph H' satisfy $|e'_j| = 2$, the hypergraph Laplacian simplifies to the graph Laplacian. Consequently, the SHGNN convolution operation reduces to the GCN operation. □

3.2 Algorithm for SuperHypergraph Neural Network (SHGNN)

We present a detailed algorithm for implementing the SuperHypergraph Neural Network (SHGNN), along with an analysis of its time and space complexity. The algorithm is described below.

Algorithm 1: SuperHypergraph Neural Network Convolution

Input:

- SuperHyperGraph $H = (V, E)$ with base vertices V_0 (where $|V_0| = n$);
- Feature matrix $X \in \mathbb{R}^{n \times d}$;
- Hyperedge weights $w(e'_j)$ for each hyperedge $e'_j \in E'$;
- Weight matrix $\Theta \in \mathbb{R}^{d \times c}$;
- Activation function σ.

Output: Output feature matrix $Y \in \mathbb{R}^{n \times c}$

1. Expand SuperHyperGraph to obtain Expanded Hypergraph $H' = (V_0, E')$;
foreach *superedge* $e \in E$ **do**
  $e' \leftarrow \bigcup_{v \in e} v$; `// Expand to base vertices`
  Add e' to E';
end
2. Construct incidence matrix $H' \in \mathbb{R}^{n \times m}$, where $m = |E'|$;
Initialize H' as a sparse zero matrix;
for $j \leftarrow 1$ **to** m **do**
  foreach *vertex* $v_i \in e'_j$ **do**
    $H'_{ij} \leftarrow 1$;
  end
end
3. Compute vertex degrees D_V;
for $i \leftarrow 1$ **to** n **do**
  $d_V(v_i) \leftarrow \sum_{j=1}^{m} H'_{ij} \cdot w(e'_j)$;
  $(D_V)_{ii} \leftarrow d_V(v_i)$;
end
4. Compute hyperedge degrees D_E;
for $j \leftarrow 1$ **to** m **do**
  $d_E(e'_j) \leftarrow \sum_{i=1}^{n} H'_{ij}$;
  $(D_E)_{jj} \leftarrow d_E(e'_j)$;
end
5. Normalize incidence matrix $\tilde{H}$;
Compute $D_V^{-1/2}$ and D_E^{-1} (diagonal matrices);
foreach *non-zero element* H'_{ij} **do**
  $\tilde{H}_{ij} \leftarrow (D_V^{-1/2})_{ii} \cdot H'_{ij} \cdot w(e'_j) \cdot (D_E^{-1})_{jj}$;
end
6. Compute intermediate matrix M;
Compute $S \leftarrow H'^{\top} D_V^{-1/2} X$; `// Sparse matrix multiplication`
Compute $M \leftarrow \tilde{H} \cdot S$; `// Sparse matrix multiplication`
7. Compute output features Y;
$Y \leftarrow \sigma(M \cdot \Theta)$;
return Y;

Theorem 3.5. *Given a SuperHyperGraph $H = (V, E)$, base vertices V_0, feature matrix X, weight matrix Θ, and activation function σ, the algorithm computes the output feature matrix Y according to the SHGNN convolution operation:*

$$Y = \sigma\left(D_V^{-1/2} H' W D_E^{-1} H'^{\top} D_V^{-1/2} X \Theta\right),$$

where H' is the incidence matrix of the Expanded Hypergraph $H' = (V_0, E')$, D_V and D_E are the vertex and hyperedge degree matrices, and W is the diagonal matrix of hyperedge weights.

Proof. The algorithm follows the steps required to compute the SHGNN convolution operation:

1. *Expansion to H'*: The algorithm correctly expands each superedge $e \in E$ into a hyperedge $e' \in E'$ by taking the union of all base vertices in the supervertices of e. This ensures that H' accurately represents the Expanded Hypergraph.
2. *Construction of H'*: By iterating over each hyperedge e'_j and setting $H'_{ij} = 1$ for all $v_i \in e'_j$, the incidence matrix H' is correctly constructed.
3. *Degree Matrices D_V and D_E*: The degrees are computed as per their definitions:

$$d_V(v_i) = \sum_{j=1}^{m} H'_{ij} \cdot w(e'_j), \quad d_E(e'_j) = \sum_{i=1}^{n} H'_{ij}.$$

   The diagonal matrices D_V and D_E are correctly populated with these degrees.
4. *Normalization and Computation of $\tilde{H}$*: The normalized incidence matrix $\tilde{H}$ is computed using the degrees and weights, matching the formula:

$$\tilde{H}_{ij} = (D_V^{-1/2})_{ii} \cdot H'_{ij} \cdot w(e'_j) \cdot (D_E^{-1})_{jj}.$$

5. *Convolution Operation*: The algorithm computes:

$$Y = \sigma\left(\tilde{H} \cdot H'^{\top} D_V^{-1/2} X\Theta\right),$$

   which simplifies to:

$$Y = \sigma\left(D_V^{-1/2} H' W D_E^{-1} H'^{\top} D_V^{-1/2} X\Theta\right),$$

   as per the SHGNN convolution definition.
6. *Activation Function*: The application of σ ensures the non-linear transformation is applied to the output.

Thus, each step of the algorithm correctly implements the corresponding mathematical operation in the SHGNN convolution, ensuring correctness. □

Theorem 3.6. *Let $n = |V_0|$ be the number of base vertices, $m = |E'|$ be the number of hyperedges in the Expanded Hypergraph, d be the input feature dimension, c be the output feature dimension, and $nnz(H')$ be the number of non-zero entries in the incidence matrix H'. The time complexity of the algorithm is:*

$$O\left(|E| \cdot k \cdot s + nnz(H') \cdot (d+1) + n \cdot d \cdot c\right),$$

where k is the average number of supervertices per superedge, and s is the average size of a supervertex.

Proof. We analyze the time complexity of each step in the algorithm:

1. *Expansion to H'*:
   - For each superedge $e \in E$, the expansion $e' = \bigcup_{v \in e} v$ involves $O(ks)$ operations, where k is the average number of supervertices in e, and s is the average size of a supervertex.
   - Total time for this step: $O(|E| \cdot k \cdot s)$.
2. *Construction of H'*:
   - For each hyperedge e'_j, we iterate over its vertices $v_i \in e'_j$ and set $H'_{ij} = 1$.
   - Time complexity: $O(\text{nnz}(H'))$.
3. *Compute D_V*:
   - For each vertex v_i, sum over hyperedges where $H'_{ij} = 1$.

- Time complexity: $O(\text{nnz}(H'))$.

4. *Compute* D_E:
   - For each hyperedge e'_j, sum over vertices where $H'_{ij} = 1$.
   - Time complexity: $O(\text{nnz}(H'))$.
5. *Normalize* $\tilde{H}$:
   - Multiplying diagonal matrices and updating non-zero entries.
   - Time complexity: $O(\text{nnz}(H'))$.
6. *Compute* $S = H'^{\top} D_V^{-1/2} X$:
   - Sparse matrix-vector multiplication.
   - Time complexity: $O(\text{nnz}(H') \cdot d)$.
7. *Compute* $M = \tilde{H} \cdot S$:
   - Sparse matrix-vector multiplication.
   - Time complexity: $O(\text{nnz}(H') \cdot d)$.
8. *Compute* $Y = \sigma(M \cdot \Theta)$:
   - Dense matrix multiplication: $O(n \cdot d \cdot c)$.
   - Activation function application: $O(n \cdot c)$.

   Adding up the time complexities:

$$O\left(|E| \cdot k \cdot s + \text{nnz}(H') \cdot (1 + d) + n \cdot d \cdot c\right).$$

   Thus, the time complexity of the algorithm is as stated. □

Theorem 3.7. *The space complexity of the algorithm is:*

$$O\left(nnz(H') + n \cdot (d + c) + m \cdot d + d \cdot c\right),$$

where n, m, d, c, and $nnz(H')$ *are as previously defined.*

Proof. We account for the space used by the algorithm:

1. *Incidence Matrix* H':
   - Stored in sparse format.
   - Space complexity: $O(\text{nnz}(H'))$.
2. *Degree Matrices* D_V *and* D_E:
   - Diagonal matrices.
   - Space complexity: $O(n + m)$.
3. *Feature Matrix* X:
   - Space complexity: $O(n \cdot d)$.
4. *Weight Matrix* Θ:
   - Space complexity: $O(d \cdot c)$.
5. *Intermediate Matrices* S *and* M:
   - $S \in \mathbb{R}^{m \times d}$: $O(m \cdot d)$.
   - $M \in \mathbb{R}^{n \times d}$: $O(n \cdot d)$.
6. *Output Matrix* Y:

- Space complexity: $O(n \cdot c)$.

Adding up the space complexities:

$$O\left(\text{nnz}(H') + n + m + n \cdot d + m \cdot d + n \cdot c + d \cdot c\right).$$

Simplifying, and noting that $n + m$ is dominated by $n \cdot d$ and $m \cdot d$, we have:

$$O\left(\text{nnz}(H') + n \cdot (d + c) + m \cdot d + d \cdot c\right).$$

Thus, the space complexity is as stated. □

Theorem 3.8. *If the Expanded Hypergraph H' is sparse, i.e., $nnz(H') = O(n)$, then the algorithm operates in linear time and space with respect to the number of vertices n.*

Proof. When H' is sparse, $\text{nnz}(H') = O(n)$. Substituting this into the time and space complexities:

Time Complexity:

$$O\left(|E| \cdot k \cdot s + n \cdot (d + 1) + n \cdot d \cdot c\right).$$

If $|E| \cdot k \cdot s = O(n)$ (which holds if the average superedge and supervertex sizes are bounded), the total time complexity becomes $O(n \cdot d \cdot c)$.

Space Complexity:

$$O\left(n + n \cdot (d + c) + n \cdot d + d \cdot c\right) = O\left(n \cdot (d + c) + d \cdot c\right).$$

Thus, both time and space complexities are linear in n when H' is sparse and superedge/supervertex sizes are bounded. □

3.3 n-SuperHyperGraph Neural Network

A SuperHyperGraph can be generalized to an n-SuperHyperGraph. This is defined based on the concept of the n-th powerset. The formal definition is provided below.

Definition 3.9 (Power Set)**.** (cf.[97]) Let S be a set. The *power set* of S, denoted by $\mathcal{P}(S)$, is defined as the set of all subsets of S, including the empty set and S itself. Formally, we write:

$$\mathcal{P}(S) = \{T \mid T \subseteq S\}.$$

The power set $\mathcal{P}(S)$ contains $2^{|S|}$ elements, where $|S|$ represents the cardinality of S. This is because each element of S can either be included in or excluded from each subset.

Definition 3.10 (n-th PowerSet (Recall))**.** (cf.[340, 352]) Let H be a set representing a system or structure, such as a set of items, a company, an institution, a country, or a region. The *n-th PowerSet*, denoted as $\mathcal{P}_n^*(H)$, describes a hierarchical organization of H into subsystems, sub-subsystems, and so forth. It is defined recursively as follows:

1. *Base Case:*
$$\mathcal{P}_0^*(H) := H.$$
2. *First-Level PowerSet:*
$$\mathcal{P}_1^*(H) = \mathcal{P}(H),$$
where $\mathcal{P}(H)$ is the power set of H.
3. *Higher Levels:* For $n \geq 2$, the n-th PowerSet is defined recursively as:
$$\mathcal{P}_n^*(H) = \mathcal{P}(\mathcal{P}_{n-1}^*(H)).$$

Thus, $\mathcal{P}_n^*(H)$ represents a nested hierarchy, where the power set operation $\mathcal{P}$ is applied n times. Formally:

$$\mathcal{P}_n^*(H) = \mathcal{P}(\mathcal{P}(\cdots \mathcal{P}(H) \cdots)),$$

where the power set operation $\mathcal{P}$ is repeated n times.

Example 3.11 (n-th PowerSet of a Simple Set)**.** Let $H = \{a, b\}$ be a set. The computation of $\mathcal{P}_n^*(H)$ for different n is as follows:

1. *Base Case ($n = 0$):*
$$\mathcal{P}_0^*(H) = H = \{a, b\}.$$
2. *First-Level PowerSet ($n = 1$):*
$$\mathcal{P}_1^*(H) = \mathcal{P}(H) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}.$$
3. *Second-Level PowerSet ($n = 2$):*
$$\mathcal{P}_2^*(H) = \mathcal{P}(\mathcal{P}(H)) = \mathcal{P}\left(\{\emptyset, \{a\}, \{b\}, \{a, b\}\}\right).$$
The elements of $\mathcal{P}_2^*(H)$ are all subsets of $\mathcal{P}(H)$, such as:
$$\mathcal{P}_2^*(H) = \{\emptyset, \{\emptyset\}, \{\{a\}\}, \{\{b\}\}, \{\{a, b\}\}, \{\emptyset, \{a\}\}, \ldots, \{\emptyset, \{a\}, \{b\}, \{a, b\}\}\}.$$
4. *Third-Level PowerSet ($n = 3$):*
$$\mathcal{P}_3^*(H) = \mathcal{P}(\mathcal{P}_2^*(H)).$$
The elements of $\mathcal{P}_3^*(H)$ are all subsets of $\mathcal{P}_2^*(H)$, forming a higher-order hierarchy.

This process illustrates how the n-th PowerSet recursively expands the original set H into increasingly complex hierarchical structures.

Theorem 3.12. *The n-th power set generalizes the power set.*

Proof. This is evident. □

Definition 3.13 (n-SuperHyperGraph)**.** (cf.[340]) Let V_0 be a finite set of base vertices. Define the n-th iterated power set of V_0 recursively as:
$$\mathcal{P}^0(V_0) = V_0, \quad \mathcal{P}^{k+1}(V_0) = \mathcal{P}\left(\mathcal{P}^k(V_0)\right),$$
where $\mathcal{P}(A)$ denotes the power set of set A.

An *n-SuperHyperGraph* is an ordered pair $H = (V, E)$, where:

- $V \subseteq \mathcal{P}^n(V_0)$ is the set of *supervertices*, which are elements of the n-th power set of V_0.
- $E \subseteq \mathcal{P}^n(V_0)$ is the set of *superedges*, also elements of $\mathcal{P}^n(V_0)$.

Each supervertex $v \in V$ can be:

- A single vertex ($v \in V_0$),
- A subset of V_0 ($v \subseteq V_0$),
- A subset of subsets of V_0, up to n levels ($v \in \mathcal{P}^n(V_0)$),
- An indeterminate or fuzzy set(cf.[430]),
- The null set ($v = \emptyset$).

Each superedge $e \in E$ connects supervertices, potentially at different hierarchical levels up to n.

Theorem 3.14. *[126] An n-SuperHyperGraph can generalize a superhypergraph.*

Proof. This follows directly from the definition. Refer to [126] as needed for further details. □

Corollary 3.15. *An n-SuperHyperGraph generalizes both hypergraphs and classical graphs.*

Proof. The result follows directly. □

Theorem 3.16. *[126] An n-SuperHyperGraph has a structure based on the n-th PowerSet.*

Proof. This follows directly from the definition. Refer to [126] as needed for further details. □

Definition 3.17 (Expanded Hypergraph for n-SuperHyperGraph)**.** Given an n-SuperHyperGraph $H = (V, E)$, the *Expanded Hypergraph* $H' = (V_0, E')$ is defined as follows:

- The vertex set is V_0, the base vertices.
- For each superedge $e \in E$, the corresponding hyperedge $e' \in E'$ is defined by recursively expanding all elements to base vertices:
$$e' = \text{Expand}(e) = \bigcup_{v \in e} \text{Expand}(v),$$
where the expansion function Expand is defined recursively:
$$\text{Expand}(v) = \begin{cases} \{v\}, & \text{if } v \in V_0, \\ \bigcup_{u \in v} \text{Expand}(u), & \text{if } v \subseteq \mathcal{P}^k(V_0),\ k \leq n. \end{cases}$$

Theorem 3.18. *The Expanded Hypergraph for an n-SuperHyperGraph generalizes the Expanded Hypergraph of a SuperHyperGraph.*

Proof. Let $H = (V, E)$ be an n-SuperHyperGraph and $H' = (V_0, E')$ its Expanded Hypergraph, where V_0 represents the base vertices. By definition, for each superedge $e \in E$, the corresponding hyperedge $e' \in E'$ is obtained through recursive expansion of all elements in e to base vertices using the function Expand.

If H is a SuperHyperGraph (i.e., $n = 1$), each supervertex $v \in e$ is either a base vertex or a subset of base vertices. Thus, the expansion process simplifies to:
$$e' = \bigcup_{v \in e} v,$$
which matches the definition of the Expanded Hypergraph for a SuperHyperGraph.

For $n > 1$, the recursive nature of Expand allows the expansion of n-nested supervertices into base vertices. This generalization accommodates the additional levels of nesting present in n-SuperHyperGraphs, ensuring the resulting hyperedges e' in H' are consistent with the definition of an Expanded Hypergraph.

Hence, the definition of the Expanded Hypergraph for n-SuperHyperGraphs subsumes that for SuperHyperGraphs, making it a generalization. □

We consider the following network.

Definition 3.19 (Network for n-SuperHyperGraph)**.** Let $X \in \mathbb{R}^{|V_0| \times d}$ be the feature matrix for base vertices V_0, where $x_i \in \mathbb{R}^d$ is the feature vector of vertex $v_i \in V_0$.

Define the incidence matrix $H' \in \mathbb{R}^{|V_0| \times |E'|}$ of the Expanded Hypergraph H' by:
$$H'_{ij} = \begin{cases} 1, & \text{if } v_i \in e'_j, \\ 0, & \text{otherwise.} \end{cases}$$

Define the diagonal vertex degree matrix $D_V \in \mathbb{R}^{|V_0| \times |V_0|}$ and hyperedge degree matrix $D_E \in \mathbb{R}^{|E'| \times |E'|}$ by:
$$(D_V)_{ii} = d_V(v_i) = \sum_{j=1}^{|E'|} H'_{ij} w(e'_j),$$
$$(D_E)_{jj} = d_E(e'_j) = \sum_{i=1}^{|V_0|} H'_{ij}.$$

Here, $w(e'_j)$ is the weight assigned to hyperedge e'_j.

The convolution operation in the n-SHGNN is defined as:
$$Y = \sigma\left(D_V^{-1/2} H' W D_E^{-1} H'^{\top} D_V^{-1/2} X \Theta\right),$$
where:

- $Y \in \mathbb{R}^{|V_0| \times c}$ is the output feature matrix.
- $W \in \mathbb{R}^{|E'| \times |E'|}$ is the diagonal matrix of hyperedge weights.
- $\Theta \in \mathbb{R}^{d \times c}$ is the learnable weight matrix.
- σ is an activation function (e.g., ReLU[175]).

Theorem 3.20. *The SuperHyperGraph Neural Network (SHGNN) is a special case of the n-SHGNN when $n = 1$.*

Proof. When $n = 1$, the n-SuperHyperGraph reduces to a standard SuperHyperGraph:

$$V \subseteq \mathcal{P}(V_0), \quad E \subseteq \mathcal{P}(V).$$

The expansion operation simplifies to:

$$\text{Expand}(v) = \begin{cases} \{v\}, & \text{if } v \in V_0, \\ v, & \text{if } v \subseteq V_0. \end{cases}$$

Thus, the definitions and algorithms of n-SHGNN coincide with those of SHGNN. Therefore, SHGNN is a special case of n-SHGNN when $n = 1$. □

As algorithms for n-SuperHyperGraphs, the following two algorithms are considered.

Algorithm 2: Expanded Hypergraph Construction

Input: An n-SuperHyperGraph $H = (V, E)$
Output: Expanded Hypergraph $H' = (V_0, E')$

Initialize $E' = \emptyset$;
foreach *superedge* $e \in E$ **do**
  $e' \leftarrow \text{Expand}(e)$;
  Add e' to E';
end
return $H' = (V_0, E')$;

Algorithm 3: n-SHGNN Convolution Operation

Input:
- Feature matrix $X \in \mathbb{R}^{|V_0| \times d}$.
- Expanded Hypergraph $H' = (V_0, E')$.
- Hyperedge weight matrix W.
- Learnable weight matrix Θ.
- Activation function σ.

Output: Output feature matrix $Y \in \mathbb{R}^{|V_0| \times c}$

Compute incidence matrix H';
Compute degree matrices D_V and D_E;
Normalize matrices: $\hat{H} = D_V^{-1/2} H' W D_E^{-1}$;
Compute $Y = \sigma\left(\hat{H} H'^{\top} D_V^{-1/2} X \Theta\right)$;
return Y;

Theorem 3.21. *The n-SHGNN convolution algorithm correctly computes the output feature matrix Y as per the convolution operation defined for n-SuperHyperGraphs.*

Proof. The algorithm follows the steps of the convolution operation:

1. Constructs the Expanded Hypergraph H' by expanding superedges e to base vertices V_0.
2. Computes the incidence matrix H' accurately.

3. Calculates degree matrices D_V and D_E according to their definitions.
4. Performs normalization and computes $\hat{H}$.
5. Computes the convolution $Y = \sigma\left(\hat{H}H'^{\top}D_V^{-1/2}X\Theta\right)$.

Each step adheres to the mathematical definitions, ensuring correctness. □

Theorem 3.22. *Let $N = |V_0|$, $M = |E|$, d be the feature dimension, c be the output dimension, and k be the maximum size of expanded hyperedges. The time complexity of the n-SHGNN convolution algorithm is $O(Mk^n + Ndc)$.*

Proof. We examine the complexity of each step in the algorithm.

- *Expanded Hypergraph Construction*:
  - For each superedge e, Expand(e) may involve up to k^n operations.
  - Total time: $O(Mk^n)$.
- *Incidence Matrix Computation*:
  - Time proportional to the number of non-zero entries: $O(Nk^n)$.
- *Degree Matrices and Normalization*:
  - Time: $O(N + |E'|)$.
- *Convolution Computation*:
  - Matrix multiplications involving sparse matrices.
  - Time: $O(Ndc)$.

Total time complexity is dominated by $O(Mk^n + Ndc)$. □

Theorem 3.23. *The space complexity of the n-SHGNN convolution algorithm is $O(Nk^n + Nd + Nc)$.*

Proof. We examine the complexity of each step in the algorithm.

- *Incidence Matrix H'*:
  - Space: $O(Nk^n)$.
- *Degree Matrices*:
  - Space: $O(N + |E'|)$.
- *Feature Matrices*:
  - Input X: $O(Nd)$.
  - Output Y: $O(Nc)$.

Total space complexity is $O(Nk^n + Nd + Nc)$. □

3.4 Dynamic Superhypergraph Neural Network

In this subsection, we define the Dynamic Superhypergraph Neural Network, building upon the concept of the Dynamic Hypergraph Neural Network [204]. A Dynamic Hypergraph Neural Network models evolving relationships within hypergraphs, learning from time-varying node and hyperedge interactions to facilitate dynamic data analysis (cf. [172, 210, 240, 395, 400, 454]). The Dynamic Hypergraph Neural Network can also be viewed as an extension of dynamic graph neural networks[118, 159, 237, 361] to the domain of hypergraphs. The definitions and theorems of related concepts are provided below.

Definition 3.24 (Dynamic Hypergraph)**.** [204] A *Dynamic Hypergraph* at layer l is represented as $H_l = (V, E_l)$, where:

- V is the set of vertices corresponding to data samples.
- E_l is the set of hyperedges at layer l, dynamically constructed based on the feature embeddings X_l of the vertices at layer l.

Hyperedges in E_l are constructed using clustering or nearest-neighbor methods to capture local and global relationships among vertices.

Definition 3.25 (Dynamic Hypergraph Neural Network (DHGNN))**.** [204] A *Dynamic Hypergraph Neural Network (DHGNN)* is a neural network architecture where each layer l consists of:

- *Dynamic Hypergraph Construction (DHG)*: Updates the hypergraph $H_l = (V, E_l)$ based on the feature embeddings X_l from the previous layer.
- *Hypergraph Convolution (HGC)*: Performs feature aggregation from vertices to hyperedges and vice versa to produce updated embeddings X_{l+1}.

The output of the l-th layer is:

$$X_{l+1} = \sigma\left(W_l X_l + \mathrm{HGC}(H_l, X_l)\right),$$

where W_l is a learnable weight matrix and σ is an activation function.

Definition 3.26. A *Dynamic SuperHypergraph* is a sequence of n-SuperHyperGraphs $\{H^{(l)} = (V^{(l)}, E^{(l)})\}_{l=0}^{L}$, where each layer l represents a SuperHyperGraph at a specific time or iteration, and:

- $V^{(l)} \subseteq \mathcal{P}^n(V_0)$ is the set of supervertices at layer l, where V_0 is the base set of vertices, and $\mathcal{P}^n(V_0)$ is the n-th iterated power set of V_0.
- $E^{(l)} \subseteq \mathcal{P}^n(V_0)$ is the set of superedges at layer l.

The evolution of the SuperHyperGraph from layer l to $l + 1$ may depend on the features or embeddings of the supervertices at layer l.

Theorem 3.27. *A Dynamic SuperHypergraph $\{H^{(l)} = (V^{(l)}, E^{(l)})\}_{l=0}^{L}$ generalizes the concept of a SuperHyperGraph $H = (V, E)$, as:*

1. *Each static layer $H^{(l)}$ is a valid SuperHyperGraph.*
2. *The sequence of layers allows for dynamic evolution, which extends the static structure of a single SuperHyperGraph to include temporal or iterative dynamics.*

Proof. We prove this theorem in two steps:

1. Static Layer Correspondence: By definition, each layer $H^{(l)} = (V^{(l)}, E^{(l)})$ satisfies the properties of a SuperHyperGraph:

- $V^{(l)} \subseteq \mathcal{P}^n(V_0)$, ensuring that the vertices are subsets of the n-th iterated power set of the base vertex set V_0.
- $E^{(l)} \subseteq \mathcal{P}^n(V_0)$, ensuring that the edges connect subsets of $V^{(l)}$.

Thus, each individual $H^{(l)}$ is a valid SuperHyperGraph.

2. Dynamic Evolution: In a Dynamic SuperHypergraph, the evolution from layer l to $l+1$ is governed by transformations applied to the supervertices or superedges. These transformations can be defined using feature propagation, embedding updates, or external conditions. This dynamic evolution introduces a temporal or iterative dimension to the SuperHyperGraph structure, which cannot be captured by a static SuperHyperGraph.

A SuperHyperGraph $H = (V, E)$ can be viewed as a special case of a Dynamic SuperHypergraph where all layers $H^{(l)}$ are identical for $l = 0, \ldots, L$, and no evolution occurs between layers.

The Dynamic SuperHypergraph $\{H^{(l)}\}$ generalizes the static SuperHyperGraph H by adding a layer-wise temporal or iterative structure. □

Theorem 3.28. *A Dynamic SuperHypergraph generalizes a Dynamic Hypergraph.*

Proof. A Dynamic Hypergraph is a special case of a Dynamic SuperHypergraph when $n = 0$ or when the supervertices are simply the base vertices V_0.

In a Dynamic Hypergraph, at each layer l, we have a hypergraph $H^{(l)} = (V, E^{(l)})$, where V is a fixed set of vertices, and $E^{(l)}$ is the set of hyperedges at layer l.

In a Dynamic SuperHypergraph, when we set $n = 0$ and $V^{(l)} = V_0$ for all l, the supervertices reduce to the base vertices, and the structure becomes a sequence of hypergraphs $\{H^{(l)} = (V_0, E^{(l)})\}$, which is exactly a Dynamic Hypergraph.

Therefore, Dynamic SuperHypergraphs generalize Dynamic Hypergraphs. □

Definition 3.29 (Dynamic SuperHypergraph Neural Network (DSHGNN))**.** A *Dynamic SuperHypergraph Neural Network (DSHGNN)* is a neural network where at each layer l, a new SuperHyperGraph $H^{(l)} = (V^{(l)}, E^{(l)})$ is constructed based on the feature embeddings $X^{(l)}$ at that layer. The DSHGNN performs convolution operations on these dynamically constructed superhypergraphs.

Specifically, the output of layer l is given by:

$$X^{(l+1)} = \sigma\left(D_V^{(l)\,-1/2} H'^{(l)} W^{(l)} D_E^{(l)\,-1} H'^{(l)\,\top} D_V^{(l)\,-1/2} X^{(l)} \Theta^{(l)}\right),$$

where:

- $H^{(l)} = (V^{(l)}, E^{(l)})$ is the SuperHyperGraph at layer l.
- $H'^{(l)}$ is the incidence matrix of the Expanded Hypergraph $H'^{(l)} = (V_0, E'^{(l)})$.
- $D_V^{(l)}$ and $D_E^{(l)}$ are the degree matrices at layer l.
- $W^{(l)}$ is the diagonal hyperedge weight matrix at layer l.
- $\Theta^{(l)}$ is the learnable weight matrix at layer l.
- σ is an activation function.

Theorem 3.30. *A Dynamic SuperHypergraph Neural Network has the structure of a Dynamic SuperHypergraph.*

Proof. In a Dynamic SuperHypergraph Neural Network, at each layer l, a new SuperHyperGraph $H^{(l)} = (V^{(l)}, E^{(l)})$ is constructed based on the embeddings $X^{(l)}$. The network updates the embeddings $X^{(l)}$ by performing operations that involve the structure of $H^{(l)}$.

Since the sequence of superhypergraphs $\{H^{(l)}\}$ evolves over the layers of the network, and each $H^{(l)}$ is a SuperHyperGraph, the network inherently operates on a Dynamic SuperHypergraph.

Therefore, the Dynamic SuperHypergraph Neural Network has the structure of a Dynamic SuperHypergraph. □

We present the algorithm for dynamically constructing the superhypergraph at each layer based on the current feature embeddings.

Algorithm 4: Dynamic SuperHypergraph Construction (DSHC) at Layer l

Input:

- Current feature embeddings $X^{(l)} \in \mathbb{R}^{|V_0| \times d}$.
- Parameters: number of supervertices s, supervertex size k, number of superedges t, superedge size m.

Output: Dynamic SuperHyperGraph $H^{(l)} = (V^{(l)}, E^{(l)})$.

1. Construct Supervertices;
Perform clustering (e.g., k-means) on $X^{(l)}$ to obtain s clusters;
For each cluster c_i, form a supervertex $v_i = \{v_j \in V_0 \mid v_j \text{ belongs to } c_i\}$;
Set $V^{(l)} = \{v_1, v_2, \ldots, v_s\}$;
2. Construct Superedges;
Perform higher-level clustering or grouping on supervertices to form t superedges;
For each group g_i, form a superedge $e_i = \{v_j \in V^{(l)} \mid v_j \text{ belongs to } g_i\}$;
Set $E^{(l)} = \{e_1, e_2, \ldots, e_t\}$;
return $H^{(l)} = (V^{(l)}, E^{(l)})$;

Theorem 3.31. *The DSHGNN algorithm computes the feature embeddings $X^{(l+1)}$ at each layer l correctly according to the convolution operation defined for the dynamically constructed superhypergraph $H^{(l)}$.*

Proof. The DSHGNN algorithm follows these steps:

1. *Dynamic SuperHypergraph Construction*: The algorithm constructs $H^{(l)}$ based on $X^{(l)}$, ensuring that the supervertices $V^{(l)}$ and superedges $E^{(l)}$ capture the relationships inherent in the current feature embeddings.
2. *Expanded Hypergraph Construction*: The Expanded Hypergraph $H'^{(l)}$ accurately reflects the connections between base vertices V_0 through the supervertices and superedges in $H^{(l)}$.
3. *Incidence Matrix and Degree Matrices*: The incidence matrix $H'^{(l)}$ and the degree matrices $D_V^{(l)}$ and $D_E^{(l)}$ are computed correctly as per the definitions.
4. *Convolution Operation*: The convolution operation is performed exactly as defined, applying the appropriate normalization and combining the feature embeddings with the learnable parameters $\Theta^{(l)}$.
5. *Activation Function*: The non-linear activation σ is applied to introduce non-linearity.

Thus, the algorithm correctly implements the DSHGNN convolution operation, ensuring that $X^{(l+1)}$ is computed accurately at each layer. □

Theorem 3.32. *Let $n = |V_0|$ be the number of base vertices, s be the number of supervertices, t be the number of superedges, d be the feature dimension, and c be the output dimension. The time complexity of the DSHGNN algorithm at each layer is:*

$$O\left(ndk + sdk + tsk + nc\right),$$

where k is the average size of supervertices and superedges.

Proof. We analyze the time complexity step by step.

Dynamic SuperHypergraph Construction

- Clustering to form supervertices: $O(nd)$ (e.g., k-means clustering).
- Forming superedges from supervertices: $O(sd)$ (clustering supervertices).

Expanded Hypergraph Construction

- For each superedge e, forming $e' = \bigcup_{v \in e} v$: $O(k^2)$ per superedge, assuming k is the average size of v and e.
- Total time: $O(tsk)$.

Convolution Operation

- Multiplications involving sparse matrices $H'^{(l)}$: $O(\text{nnz}(H'^{(l)})d)$.
- Since $\text{nnz}(H'^{(l)}) \approx nk$, total time: $O(ndk)$.

Total Time Complexity Combining the above:

$$O\left(nd + sd + tsk + ndk + nc\right) = O\left(ndk + sdk + tsk + nc\right).$$

Assuming s, t, and k are much smaller than n, the dominant term is $O(ndk)$. □

Theorem 3.33. *The space complexity of the DSHGNN algorithm at each layer is:*

$$O\left(nd + sd + nnz(H'^{(l)}) + dc\right),$$

where $nnz(H'^{(l)})$ is the number of non-zero entries in the incidence matrix $H'^{(l)}$.

Proof. We account for the space used:

- Feature embeddings $X^{(l)}$ and $X^{(l+1)}$: $O(nd)$.
- Supervertices and their embeddings: $O(sd)$.
- Incidence matrix $H'^{(l)}$: $O(\text{nnz}(H'^{(l)}))$.
- Weight matrices $\Theta^{(l)}$: $O(dc)$.

Total space complexity:

$$O\left(nd + sd + \text{nnz}(H'^{(l)}) + dc\right).$$

□

Theorem 3.34. *The Dynamic Hypergraph Neural Network (DHGNN) is a special case of the Dynamic Super-Hypergraph Neural Network (DSHGNN). Specifically, when all supervertices in DSHGNN are singleton subsets of V_0 (i.e., $\forall v \in V^{(l)}$, $v = \{v_i\}$ for some $v_i \in V_0$), the DSHGNN reduces to the DHGNN.*

Proof. When all supervertices are singletons:

$$V^{(l)} = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}.$$

Each supervertex corresponds directly to a base vertex in V_0. The superedges $E^{(l)}$ then connect these singleton supervertices, effectively becoming hyperedges over V_0.

The Expanded Hypergraph $H'^{(l)}$ has hyperedges e' formed as:

$$e' = \bigcup_{v \in e} v = \bigcup_{v \in e} \{v_i\} = \{v_i \mid v_i \in e\}.$$

Thus, the Expanded Hypergraph $H'^{(l)}$ is identical to the hypergraph used in DHGNN at layer l.

The convolution operation in DSHGNN becomes:

$$X^{(l+1)} = \sigma\left(D_V^{(l)\,-1/2} H'^{(l)} W^{(l)} D_E^{(l)\,-1} H'^{(l)\,\top} D_V^{(l)\,-1/2} X^{(l)} \Theta^{(l)}\right),$$

which matches the convolution operation in DHGNN.

Therefore, DSHGNN reduces to DHGNN when supervertices are singletons, proving that DSHGNN generalizes DHGNN. □

3.5 Multi-Graph Neural Networks and Their Generalization

Multi-Graph Neural Networks have been proposed in recent years[421]. However, we demonstrate that they can be mathematically generalized within the framework of n-SuperHyperGraph Neural Networks. Below, we present the relevant definitions and theorems, including related concepts.

Definition 3.35. (cf.[57]) A *multi-graph* is a generalization of a graph that allows multiple edges, also called parallel edges, between the same pair of vertices. Formally, a multi-graph G is defined as:

$$G = (V, E, \varphi),$$

where:

- V is a finite set of vertices (nodes).
- E is a finite set of edges.
- $\varphi : E \to \{\{u, v\} \mid u, v \in V\}$ is a mapping that associates each edge $e \in E$ with an unordered pair of vertices $u, v \in V$. For directed multi-graphs, $\varphi(e)$ maps to ordered pairs (u, v).

Properties

- *Parallel Edges:* Unlike a simple graph, a multi-graph allows multiple edges between the same pair of vertices.
- *Loops:* Depending on the context, a multi-graph may also allow edges that connect a vertex to itself, called loops.
- *Representation:* Each edge e is distinguished by its unique identity in E, even if it connects the same vertices as another edge.

Theorem 3.36. *An n-SuperHyperGraph generalizes a multi-graph.*

Proof. To show that an n-SuperHyperGraph can generalize a multi-graph, we construct a mapping from a multi-graph $G = (V, E, \varphi)$ to an n-SuperHyperGraph $H = (V', E')$ and demonstrate that the operations and representations in G can be captured within H.

In the multi-graph G, the vertex set is V. In the n-SuperHyperGraph H, let the base vertex set V_0 correspond directly to V. Thus, each vertex $v \in V$ in G is represented as a supervertex $v \in V_0 \subseteq \mathcal{P}^n(V_0)$ in H.

Each edge $e \in E$ in the multi-graph G is mapped to a superedge $e' \in E'$ in H. Specifically:

$$e' = \{u, v\}, \quad \text{where } \varphi(e) = \{u, v\}, \text{ and } u, v \in V_0.$$

For parallel edges, each edge e in G is assigned a unique identity and mapped to a distinct superedge in E'. Thus, E' may contain multiple superedges connecting the same pair of vertices, replicating the parallel edge property of a multi-graph.

If G allows loops (edges connecting a vertex to itself), such edges $e \in E$ can be mapped to superedges $e' = \{v, v\}$ in H. This is valid in the n-SuperHyperGraph framework since $v \in V_0$.

For $n > 1$, the n-SuperHyperGraph structure provides additional hierarchical levels that are not utilized in the basic mapping of a multi-graph. Thus, a multi-graph is a special case of an n-SuperHyperGraph where $n \geq 1$ and all supervertices and superedges reside at the base level ($\mathcal{P}^0(V_0) = V_0$).

The construction above demonstrates that the vertex and edge structures of any multi-graph G can be faithfully represented within an n-SuperHyperGraph H. Additionally, the n-SuperHyperGraph framework supports the generalization to hierarchical and nested structures beyond what is possible in a multi-graph. Therefore, n-SuperHyperGraphs generalize multi-graphs. □

Definition 3.37. [421] A *Multi-Graph Neural Network (MGNN)* is an extension of Graph Neural Networks (GNNs) designed to operate on *multi-graphs*. In a multi-graph, multiple edges (possibly of different types) are allowed between the same pair of nodes. This structure enables the modeling of complex relationships in data where interactions can occur through various channels or modalities.

Formally, let $G = (V, E, T)$ be a multi-graph, where:

- V is the set of nodes.
- $E \subseteq V \times V \times T$ is the set of edges.

- T is the set of edge types.

Each edge $e = (u, v, t) \in E$ represents an interaction of type $t \in T$ between nodes u and v.

In an MGNN, the message passing and aggregation functions are adapted to handle multiple edge types. The node representation update typically involves aggregating messages over all edge types:

$$\mathbf{h}_v^{(t+1)} = \phi\left(\mathbf{h}_v^{(t)}, \bigoplus_{t' \in T} \bigoplus_{u \in \mathcal{N}_v^{t'}} \psi^{t'}\left(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{e}_{uv}^{t'}\right)\right),$$

where:

- $\mathbf{h}_v^{(t)}$ is the representation of node v at layer t.
- $\mathcal{N}_v^{t'}$ is the set of neighbors of node v connected via edges of type t'.
- $\psi^{t'}$ is the message function for edge type t'.
- ϕ is the node update function.
- $\bigoplus$ denotes an aggregation operator (e.g., sum, mean, max).
- $\mathbf{e}_{uv}^{t'}$ is the feature of edge (u, v, t').

Theorem 3.38. *An n-SuperHyperGraph Neural Network (n-SHGNN) can generalize a Multi-Graph Neural Network (MGNN).*

Proof. To prove this theorem, we need to demonstrate that any MGNN can be represented as a special case of an n-SHGNN for some appropriate n.

Mapping the Multi-Graph to an n-SuperHyperGraph Let $G = (V, E, T)$ be a multi-graph, where multiple edges of different types can exist between the same pair of nodes. We aim to construct an n-SuperHyperGraph $H = (V', E')$ such that the MGNN operations on G can be emulated by an n-SHGNN operating on H.

Construction of the n-SuperHyperGraph

- *Base Vertices:* Let $V_0 = V$, the original set of nodes in the multi-graph.
- *Supervertices:* For each edge type $t \in T$, define a supervertex v_t at the first level of the power set ($n = 1$):

$$v_t = \{v \in V_0 \mid v \text{ participates in at least one edge of type } t\}.$$

- *Superedges:* For each edge $e = (u, v, t) \in E$, define a superedge e' connecting the corresponding nodes and the supervertex v_t:

$$e' = \{u, v, v_t\}.$$

By constructing supervertices corresponding to each edge type and connecting them via superedges to the nodes involved in edges of that type, we encapsulate the multi-graph's multiple edge types within the n-SuperHyperGraph structure.

In the n-SHGNN, message passing can proceed as follows:

- Nodes exchange messages via superedges, which now represent the multi-graph's edges along with their types.
- The supervertex v_t serves as a mediator that allows nodes connected by edges of type t to share information specific to that edge type.

The MGNN's handling of multiple edge types through type-specific message functions ψ^t can be replicated in the n-SHGNN by defining superedges and supervertices that correspond to these types. The hierarchical structure of the n-SuperHyperGraph allows for the encapsulation of edge type information within the graph topology.

For more complex multi-graphs or for edge types that have hierarchical relationships, a higher n can be chosen to capture the necessary levels of nesting. However, for standard MGNNs, setting $n = 1$ suffices.

Since we can construct an n-SuperHyperGraph H such that the MGNN operations on G are equivalent to n-SHGNN operations on H, it follows that an n-SHGNN can generalize an MGNN. □

3.6 Revisiting Definitions for SHGNN

In this subsection, we revisit several definitions relevant to the SuperHyperGraph Neural Network (SHGNN). Specifically, we briefly examine concepts such as the SuperHyperGraph Laplacian, SuperHyperGraph Convolution, SuperHyperGraph Clustering, and SuperHyperGraph Degree Centrality.

3.6.1 SuperHyperGraph Laplacian

The SuperHyperGraph Laplacian can be specifically defined as follows. We prove that it generalizes the HyperGraph Laplacian. For clarity, the Graph Laplacian is a matrix representing a graph's structure, used to analyze connectivity and spectral properties (cf.[282, 438]).

Definition 3.39 (HyperGraph Laplacian)**.** (cf.[75, 137]) Define the incidence matrix $H \in \mathbb{R}^{n\times m}$ of the hypergraph $\mathcal{H}$ by:

$$H_{ij} = \begin{cases} 1, & \text{if } v_i \in e_j, \\ 0, & \text{otherwise.} \end{cases}$$

Define the diagonal *vertex degree matrix* $D_v \in \mathbb{R}^{n\times n}$ with entries:

$$(D_v)_{ii} = d_v(v_i) = \sum_{j=1}^{m} H_{ij} w(e_j),$$

where $w(e_j)$ is the weight assigned to hyperedge e_j.
Define the diagonal *hyperedge degree matrix* $D_e \in \mathbb{R}^{m\times m}$ with entries:

$$(D_e)_{jj} = d_e(e_j) = \sum_{i=1}^{n} H_{ij}.$$

The *hypergraph Laplacian* $L \in \mathbb{R}^{n\times n}$ is defined as:

$$L = I - D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2},$$

where $W \in \mathbb{R}^{m\times m}$ is the diagonal matrix of hyperedge weights $w(e_j)$, and I is the identity matrix.

Definition 3.40 (SuperHyperGraph Laplacian)**.** To define the Laplacian for a SuperHyperGraph, we construct the *Expanded Hypergraph* $H' = (V_0, E')$:

- The vertex set is V_0.
- For each superedge $e \in E$, the corresponding hyperedge $e' \in E'$ is:

$$e' = \bigcup_{v\in e} v.$$

Define the incidence matrix $H' \in \mathbb{R}^{|V_0|\times|E'|}$:

$$H'_{ij} = \begin{cases} 1, & \text{if } v_i \in e'_j, \\ 0, & \text{otherwise.} \end{cases}$$

Define the diagonal *vertex degree matrix* $D_V \in \mathbb{R}^{|V_0|\times|V_0|}$:

$$(D_V)_{ii} = d_V(v_i) = \sum_{j=1}^{|E'|} H'_{ij} w(e'_j).$$

Define the diagonal *hyperedge degree matrix* $D_E \in \mathbb{R}^{|E'|\times|E'|}$:

$$(D_E)_{jj} = d_E(e'_j) = \sum_{i=1}^{|V_0|} H'_{ij}.$$

The *SuperHyperGraph Laplacian* $L \in \mathbb{R}^{|V_0|\times|V_0|}$ is defined as:

$$L = I - D_V^{-1/2} H' W D_E^{-1} H'^\top D_V^{-1/2},$$

where W is the diagonal matrix of hyperedge weights $w(e'_j)$.

Theorem 3.41. *The SuperHyperGraph Laplacian L generalizes the hypergraph Laplacian. Specifically, when all supervertices are singleton sets (i.e., $V = V_0$), the SuperHyperGraph Laplacian reduces to the hypergraph Laplacian.*

Proof. When $V = V_0$, each supervertex $v \in V$ is a singleton set $\{v\}$. Consequently, each superedge $e \subseteq V$ corresponds directly to a hyperedge in the hypergraph $\mathcal{H} = (V, E)$.

In the Expanded Hypergraph H', each hyperedge e' is:

$$e' = \bigcup_{v \in e} v = \bigcup_{v \in e} \{v\} = e.$$

Thus, H' coincides with the incidence matrix H of the hypergraph. The degree matrices D_V and D_E become D_v and D_e of the hypergraph.

Therefore, the SuperHyperGraph Laplacian L reduces to:

$$L = I - D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2},$$

which is the hypergraph Laplacian. Hence, the SuperHyperGraph Laplacian generalizes the hypergraph Laplacian. □

3.6.2 SuperHyperGraph Convolution

Define SuperHyperGraph Convolution and examine its relationship with HyperGraph Convolution. For clarity, Graph Convolution is an operation aggregating node features and their neighbors' information, capturing graph structure for learning (cf.[390, 444, 455]).

Definition 3.42 (HyperGraph Convolution)**.** (cf.[38, 251]) In Hypergraph Neural Networks, the convolution operation aggregates information from hyperedges to vertices.

Given:

- Feature matrix $X \in \mathbb{R}^{n \times d}$, where x_i is the feature vector of vertex v_i.
- Learnable weight matrix $\Theta \in \mathbb{R}^{d \times c}$.

The hypergraph convolution is defined as:

$$Y = \sigma\left(D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2} X \Theta\right),$$

where σ is an activation function (e.g., ReLU).

Definition 3.43. Let $X \in \mathbb{R}^{|V_0| \times d}$ be the feature matrix for the base vertices V_0, where each row x_i corresponds to the feature vector of vertex $v_i \in V_0$. The convolution operation is defined as:

$$Y = \sigma\left(D_V^{-1/2} H' W D_E^{-1} H'^\top D_V^{-1/2} X \Theta\right),$$

where:

- σ is an activation function (e.g., ReLU).
- $\Theta \in \mathbb{R}^{d \times c}$ is a learnable weight matrix.
- Other matrices are as previously defined.

Theorem 3.44. *The SuperHyperGraph convolution operation generalizes the hypergraph convolution. When $V = V_0$, the SuperHyperGraph convolution reduces to the hypergraph convolution.*

Proof. With $V = V_0$ and $H' = H$, the convolution formula becomes:

$$Y = \sigma\left(D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2} X \Theta\right),$$

which is the hypergraph convolution formula. Thus, the SuperHyperGraph convolution generalizes the hypergraph convolution. □

3.6.3 SuperHyperGraph Clustering

Define SuperHyperGraph Clustering and examine its relationship with HyperGraph Clustering[67, 138, 227, 230]. Note that graph clustering partitions a graph into groups of nodes (clusters) such that nodes within the same cluster are highly connected [381, 391, 424].

Definition 3.45 (Graph Clustering)**.** (cf.[243, 280]) Let $G = (V, E, w)$ be a weighted graph, where:

- V is the set of vertices,
- $E \subseteq V \times V$ is the set of edges,
- $w : E \to \mathbb{R}^+$ assigns a positive weight to each edge.

A *clustering* of the graph G is a partition of the vertex set V into k disjoint subsets:

$$C = \{C_1, C_2, \ldots, C_k\},$$

such that:

1. $\bigcup_{i=1}^{k} C_i = V$,
2. $C_i \cap C_j = \emptyset$ for $i \neq j$.

Each subset C_i is called a *cluster*. The quality of the clustering is often measured by evaluating the edge weights within clusters (intra-cluster similarity) and between clusters (inter-cluster dissimilarity).

Example 3.46 (Clustering a Simple Graph)**.** Consider the graph $G = (V, E, w)$ with:

$$V = \{A, B, C, D, E\}, \quad E = \{(A, B), (A, C), (B, C), (B, D), (C, E)\},$$

and edge weights:

$$w(A, B) = 1, \quad w(A, C) = 2, \quad w(B, C) = 2, \quad w(B, D) = 1, \quad w(C, E) = 3.$$

A possible clustering is:

$$C_1 = \{A, B, C\}, \quad C_2 = \{D, E\}.$$

Evaluation:

- *Intra-cluster weight (within C_1):*

$$w(A, B) + w(A, C) + w(B, C) = 1 + 2 + 2 = 5.$$

- *Inter-cluster weight (between C_1 and C_2):*

$$w(B, D) + w(C, E) = 1 + 3 = 4.$$

This clustering balances high intra-cluster similarity and low inter-cluster dissimilarity, making it a good partition.

Definition 3.47 (HyperGraph Clustering)**.** (cf.[67, 138, 227, 230]) In hypergraph clustering, the goal is to partition the vertex set $\mathcal{V}$ into k clusters $\{C_1, C_2, \ldots, C_k\}$ that minimize the normalized cut:

$$\text{NCut}(C) = \sum_{i=1}^{k} \frac{\text{cut}(C_i, \overline{C_i})}{\text{vol}(C_i)},$$

where:

- $\text{cut}(C_i, \overline{C_i}) = \sum_{e \in \mathcal{E}} w(e) \frac{|e \cap C_i| \cdot |e \cap \overline{C_i}|}{|e|}$.
- $\text{vol}(C_i) = \sum_{v_j \in C_i} d_v(v_j)$.

Definition 3.48 (SuperHyperGraph clustering)**.** A *clustering* of a SuperHyperGraph $H = (V, E)$ is a partition $C = \{C_1, C_2, \ldots, C_k\}$ of the base vertex set V_0, where each cluster $C_i \subseteq V_0$.

The *normalized cut* criterion for clustering in a SuperHyperGraph is defined using the Laplacian L of the Expanded Hypergraph H'. The objective is to minimize:

$$\mathrm{NCut}(C) = \sum_{i=1}^{k} \frac{\mathrm{vol}(C_i, \overline{C_i})}{\mathrm{vol}(C_i)},$$

where:

- $\mathrm{vol}(C_i) = \sum_{v_j \in C_i} d_V(v_j)$,
- $\mathrm{vol}(C_i, \overline{C_i}) = \sum_{v_j \in C_i, v_k \in \overline{C_i}} L_{jk}$,
- $\overline{C_i} = V_0 \setminus C_i$.

Theorem 3.49. *The clustering methods for SuperHyperGraphs generalize those for hypergraphs. In particular, spectral clustering using the SuperHyperGraph Laplacian reduces to hypergraph spectral clustering when $V = V_0$.*

Proof. In hypergraph spectral clustering, the Laplacian of the hypergraph is used to compute eigenvectors corresponding to the smallest non-zero eigenvalues, which are then used to partition the vertex set V_0.

For the SuperHyperGraph, when $V = V_0$, the Laplacian L becomes the hypergraph Laplacian. Therefore, spectral clustering on the SuperHyperGraph reduces to spectral clustering on the hypergraph.

Hence, clustering methods in SuperHyperGraphs generalize those in hypergraphs. □

3.6.4 Degree Centrality in Superhypergraph

We discuss the concept of degree centrality in a superhypergraph. Degree centrality measures the importance of a node in a graph by counting the number of direct connections (edges) it has (cf.[37, 441]).

Definition 3.50 (degree centrality in hypergraph)**.** [211, 220, 397] In hypergraphs, the *degree centrality* of a vertex v_i is:

$$C(v_i) = d_v(v_i) = \sum_{j=1}^{m} H_{ij} w(e_j).$$

Definition 3.51 (degree centrality in superhypergraph)**.** The *degree centrality* of a base vertex $v_i \in V_0$ in superhypergraph is defined as:

$$C(v_i) = d_V(v_i) = \sum_{j=1}^{|E'|} H'_{ij}\, w(e'_j).$$

Theorem 3.52. *The degree centrality defined for SuperHyperGraphs generalizes the degree centrality in hypergraphs. Specifically, when $V = V_0$, the centrality measure reduces to the hypergraph degree centrality.*

Proof. When $V = V_0$, the degree centrality formula becomes:

$$C(v_i) = \sum_{j=1}^{|E|} H_{ij}\, w(e_j),$$

which is the standard degree centrality in hypergraphs.

Therefore, the SuperHyperGraph centrality measure generalizes the hypergraph centrality measure. □

3.6.5 *n*-SuperHyperGraph Attention

We provide precise mathematical definitions of Hypergraph Attention and extend it to *n*-SuperHyperGraphs, defining the *n*-SuperHyperGraph Attention mechanism. Note that graph Attention leverages attention mechanisms to dynamically weigh neighbor nodes, enhancing message-passing efficiency and representation learning in graph neural networks (cf.[61, 68, 318, 385, 398, 399]).

Definition 3.53 (Hypergraph Attention)**.** [38, 77, 103, 222, 247, 315, 394] In Hypergraph Attention, we introduce learnable attention coefficients to the incidence matrix to capture the importance of connections between vertices and hyperedges.

For each vertex v_i and hyperedge e_j, we compute an attention coefficient α_{ij} defined as:

$$\alpha_{ij} = \frac{\exp\left(\sigma\left(a^\top [x_i \,\|\, u_j]\right)\right)}{\sum_{k \in \mathcal{E}_i} \exp\left(\sigma\left(a^\top [x_i \,\|\, u_k]\right)\right)},$$

where:

- σ is a nonlinear activation function (e.g., LeakyReLU).
- $a \in \mathbb{R}^{2d'}$ is a learnable weight vector.
- $\|$ denotes vector concatenation.
- $x_i' = x_i\Theta$ and $u_j' = u_j\Theta$, where $\Theta \in \mathbb{R}^{d \times d'}$ is a shared weight matrix.
- u_j is the feature representation of hyperedge e_j, typically defined as:

$$u_j = \frac{1}{|e_j|} \sum_{v_k \in e_j} x_k.$$

- $\mathcal{E}_i = \{e_j \in \mathcal{E} \mid H_{ij} = 1\}$ is the set of hyperedges incident to vertex v_i.

The attention-based incidence matrix $\tilde{H}$ has entries $\tilde{H}_{ij} = \alpha_{ij}$.
The hypergraph attention convolution operation is then defined as:

$$X' = \sigma\left(D_v^{-1}\tilde{H}WD_e^{-1}\tilde{H}^\top X\right).$$

Definition 3.54 (*n*-SuperHyperGraph Attention)**.** In *n*-SuperHyperGraph Attention, we introduce attention coefficients between supervertices and superedges.

For each base vertex $v_i \in V_0$ and superedge $e_j' \in \mathcal{E}'^{(n)}$, we compute an attention coefficient α_{ij} as:

$$\alpha_{ij} = \frac{\exp\left(\sigma\left(a^\top [x_i \,\|\, u_j]\right)\right)}{\sum_{k \in \mathcal{E}_i} \exp\left(\sigma\left(a^\top [x_i \,\|\, u_k]\right)\right)},$$

where:

- x_i is the feature vector of base vertex v_i.
- u_j is the feature representation of superedge e_j', defined as an aggregation of features of the elements (which can be supervertices or sets thereof) in e_j'.
- $\mathcal{E}_i$ is the set of superedges incident to base vertex v_i.

The attention-based incidence matrix $\tilde{H}^{(n)}$ has entries $\tilde{H}_{ij}^{(n)} = \alpha_{ij}$.
The *n*-SuperHyperGraph attention convolution operation is defined as:

$$X' = \sigma\left(D_v^{-1}\tilde{H}^{(n)}WD_e^{-1}\tilde{H}^{(n)\top}X\right).$$

Theorem 3.55. *The n-SuperHyperGraph Attention mechanism generalizes the Hypergraph Attention mechanism. Specifically, when n = 1, the n-SuperHyperGraph Attention reduces to the standard Hypergraph Attention.*

Proof. Consider the case when $n = 1$. Then:

$$\mathcal{P}^1(V_0) = \mathcal{P}(V_0),$$

so the supervertices $\mathcal{V}^{(1)} \subseteq \mathcal{P}(V_0)$.

However, to align with the standard hypergraph setting, we consider $\mathcal{V}^{(1)} = V_0$, and $\mathcal{E}^{(1)} = \{e_j \subseteq V_0 \mid e_j \neq \emptyset\}$, which is exactly the set of hyperedges in a standard hypergraph.

In the attention mechanism, the attention coefficients α_{ij} are computed between vertices $v_i \in V_0$ and hyperedges $e_j \subseteq V_0$.

Thus, when $n = 1$, the n-SuperHyperGraph Attention reduces to the standard Hypergraph Attention mechanism.

Therefore, the n-SuperHyperGraph Attention generalizes the Hypergraph Attention. □

4 Result: Uncertain Graph Neural Networks

In this section, we explore uncertain graph networks, including Fuzzy Graph Neural Networks, Neutrosophic Graph Neural Networks, and Plithogenic Graph Neural Networks.

4.1 Neutrosophic Graph Neural Network (N-GNN)

In this subsection, we define the concept of the *Neutrosophic Graph Neural Network (N-GNN)* and demonstrate how it generalizes the Fuzzy Graph Neural Network (F-GNN). This framework extends the Fuzzy Graph Neural Network by incorporating the structure of Neutrosophic Graphs. The following sections provide the formal definitions and related theorems.

Definition 4.1 (Neutrosophic Graph Neural Network (N-GNN))**.** A Neutrosophic Graph Neural Network (N-GNN) is a graph inference model that integrates neutrosophic logic into the framework of graph neural networks to handle uncertain, indeterminate, and inconsistent data in graph-structured information. Formally, an N-GNN is defined as a quintuple:

$$\text{N-GNN} = (G, \mathcal{N}_V, \mathcal{N}_E, \mathcal{R}_N, \mathcal{D}_N),$$

where:

- $G = (V, E)$ is a graph with vertex set V and edge set E.
- $\mathcal{N}_V$ and $\mathcal{N}_E$ are the neutrosophic fuzzification functions for vertices and edges, respectively. These functions map vertex and edge attributes to neutrosophic membership triplets:

  $$\mathcal{N}_V : \mathcal{X}_V \to [0,1]^3, \quad \mathcal{N}_E : \mathcal{X}_E \to [0,1]^3,$$

  where each output is a triplet (μ_T, μ_I, μ_F) representing the degrees of truth-membership, indeterminacy-membership, and falsity-membership.
- $\mathcal{R}_N$ represents the rule layer, which encodes neutrosophic rules to aggregate neutrosophic information from neighboring nodes and edges.
- $\mathcal{D}_N$ is the neutrosophic defuzzification function, which aggregates the outputs of the rule layer to produce crisp outputs for each vertex or edge.

Definition 4.2 (Operations in N-GNN)**.** Given an input graph $G = (V, E)$ with vertex features X_V and edge features X_E, the N-GNN operates as follows:

1. *Neutrosophic Fuzzification Layer:* Each vertex $v \in V$ and edge $e \in E$ is fuzzified into neutrosophic membership triplets using membership functions:

   $$\mathcal{N}_V(v) = (\mu_T(v), \mu_I(v), \mu_F(v)), \quad \mathcal{N}_E(e) = (\mu_T(e), \mu_I(e), \mu_F(e)).$$

2. *Rule Layer:* A set of neutrosophic rules is defined to aggregate neutrosophic information. For example:

   $$\text{IF } v \text{ has } (\mu_T^v, \mu_I^v, \mu_F^v) \text{ AND } u \text{ has } (\mu_T^u, \mu_I^u, \mu_F^u) \text{ THEN } y_k = f_k(\mathcal{N}_V(v), \mathcal{N}_V(u)),$$

   where f_k is a trainable function that operates on neutrosophic membership values.

3. *Normalization Layer:* The firing strength r_k of each rule is calculated and normalized:

$$r_k = \text{Comb}\left(\mathcal{N}_V(v), \mathcal{N}_V(u)\right), \quad \hat{r}_k = \frac{r_k}{\sum_{j=1}^{K} r_j},$$

where Comb is a combination function suitable for neutrosophic logic.

4. *Defuzzification Layer:* The normalized rule outputs are aggregated to produce crisp predictions:

$$y = \sum_{k=1}^{K} \hat{r}_k \cdot f_k\left(x_v, x_u\right).$$

Definition 4.3 (Stacked N-GNN Architecture)**.** For a multi-layer N-GNN, the l-th layer is defined as:

$$H^{(l)} = \sigma\left(f_\theta^{(l)}\left(H^{(l-1)}, A\right) + H^{(l-1)}\right),$$

where:

- $H^{(l)}$ is the output of the l-th layer.
- σ is a non-linear activation function (e.g., ReLU).
- A is the adjacency matrix of the graph.
- $f_\theta^{(l)}$ is a trainable function incorporating neutrosophic operations.

The final output of the N-GNN is:

$$Y = \text{Softmax}\left(H^{(L)}\right),$$

where L is the number of layers in the N-GNN.

Theorem 4.4. *The Neutrosophic Graph Neural Network (N-GNN) generalizes the Fuzzy Graph Neural Network (F-GNN).*

Proof. In an N-GNN, each vertex and edge is associated with a neutrosophic membership triplet (μ_T, μ_I, μ_F). Consider the special case where the indeterminacy and falsity components are zero for all vertices and edges, i.e., $\mu_I(v) = 0$ and $\mu_F(v) = 0$ for all $v \in V$, and similarly for edges. Then, the neutrosophic membership reduces to the fuzzy membership:

$$\mu_T(v) = \sigma(v), \quad \forall v \in V,$$

where $\sigma(v)$ is the fuzzy membership degree in F-GNN. Under these conditions, the N-GNN operations reduce to those of the F-GNN. Therefore, the N-GNN generalizes the F-GNN. □

Theorem 4.5. *A Neutrosophic Graph Neural Network (N-GNN), as defined, has the structural properties of a Neutrosophic Graph.*

Proof. To prove this, we verify that the structure of the N-GNN satisfies the defining properties of a Neutrosophic Graph.

1. Vertices and Edges in Neutrosophic Graphs: In a Neutrosophic Graph $G = (V, E)$, each vertex $v \in V$ is associated with a triplet $\sigma(v) = (\sigma_T(v), \sigma_I(v), \sigma_F(v))$ where $\sigma_T(v), \sigma_I(v), \sigma_F(v) \in [0, 1]$ and $\sigma_T(v) + \sigma_I(v) + \sigma_F(v) \leq 3$. Similarly, each edge $e \in E$ is associated with a triplet $\mu(e) = (\mu_T(e), \mu_I(e), \mu_F(e))$ satisfying the same constraints.

In the N-GNN, the neutrosophic fuzzification layer assigns triplets to vertices and edges:

$$\mathcal{N}_V(v) = (\mu_T(v), \mu_I(v), \mu_F(v)), \quad \mathcal{N}_E(e) = (\mu_T(e), \mu_I(e), \mu_F(e)),$$

where $\mu_T, \mu_I, \mu_F \in [0, 1]$ and the sum constraint is explicitly ensured during the mapping process. Thus, the first property of a Neutrosophic Graph is satisfied.

2. Neutrosophic Membership Consistency: In a Neutrosophic Graph, the membership of an edge depends on the membership of its incident vertices. For instance:

$$\mu_T(e) \leq \min\{\sigma_T(u), \sigma_T(v)\}, \quad \mu_I(e) \leq \max\{\sigma_I(u), \sigma_I(v)\}, \quad \mu_F(e) \geq \max\{\sigma_F(u), \sigma_F(v)\},$$

for an edge $e = (u, v)$.

In the N-GNN, during the aggregation step in the rule layer, the neutrosophic membership values for edges are derived from the memberships of adjacent vertices according to neutrosophic logical rules. This ensures that edge memberships are consistent with vertex memberships, satisfying the second property.

3. Propagation of Neutrosophic Membership: A Neutrosophic Graph allows the propagation of neutrosophic properties through its structure. In the N-GNN, the rule and aggregation layers propagate vertex and edge memberships throughout the network while preserving the neutrosophic constraints.

Let $\mathcal{R}_N$ represent the rule layer and $\mathcal{A}_N$ represent the aggregation mechanism. For a vertex v, the output neutrosophic triplet at layer l is computed as:

$$\sigma^{(l)}(v) = \mathcal{A}_N\left(\{\mathcal{R}_N(\sigma^{(l-1)}(u), \mu^{(l-1)}(e)) \mid u \in \text{neighbors}(v)\}\right),$$

where $\sigma^{(l-1)}(u)$ and $\mu^{(l-1)}(e)$ represent the triplets from the previous layer. This propagation mechanism ensures that the neutrosophic graph structure is preserved across layers.

4. Defuzzification to Classical Graph Outputs: The defuzzification layer in the N-GNN converts neutrosophic triplets into crisp outputs while maintaining consistency with the original neutrosophic structure. This aligns with the final output of a Neutrosophic Graph.

Each layer of the N-GNN maintains the structure and properties of a Neutrosophic Graph. Therefore, a Neutrosophic Graph Neural Network inherently possesses the structure of a Neutrosophic Graph, as required. □

4.2 Plithogenic Graph Neural Network (P-GNN)

Next, we define the *Plithogenic Graph Neural Network (P-GNN)* and show how it generalizes both N-GNN and F-GNN.

Definition 4.6 (Plithogenic Graph Neural Network (P-GNN))**.** A Plithogenic Graph Neural Network (P-GNN) is a graph inference model that integrates plithogenic logic into the framework of graph neural networks to handle data with degrees of appurtenance and contradiction in graph-structured information. Formally, a P-GNN is defined as:

$$\text{P-GNN} = (G, \mathcal{P}_V, \mathcal{P}_E, \mathcal{R}_P, \mathcal{D}_P),$$

where:

- $G = (V, E)$ is a graph with vertex set V and edge set E.
- $\mathcal{P}_V$ and $\mathcal{P}_E$ are the plithogenic fuzzification functions for vertices and edges, respectively. These functions map vertex and edge attributes to plithogenic membership values, which include degrees of appurtenance and contradiction.
- $\mathcal{R}_P$ represents the rule layer, which encodes plithogenic rules to aggregate plithogenic information from neighboring nodes and edges.
- $\mathcal{D}_P$ is the plithogenic defuzzification function, which aggregates the outputs of the rule layer to produce crisp outputs for each vertex or edge.

Definition 4.7 (Operations in P-GNN)**.** Given an input graph $G = (V, E)$ with vertex features X_V and edge features X_E, the P-GNN operates as follows:

1. *Plithogenic Fuzzification Layer:* Each vertex $v \in V$ and edge $e \in E$ is fuzzified into plithogenic membership values using degrees of appurtenance and contradiction.
2. *Rule Layer:* A set of plithogenic rules is defined to aggregate plithogenic information. For example:

   $$\text{IF } v \text{ has DAF } \alpha_v \text{ AND } u \text{ has DAF } \alpha_u \text{ AND DCF } \delta_{vu} \text{ THEN } y_k = f_k\left(\mathcal{P}_V(v), \mathcal{P}_V(u)\right),$$

   where f_k is a trainable function that operates on plithogenic membership values.

3. *Normalization Layer:* The firing strength r_k of each rule is calculated and normalized, taking into account degrees of contradiction.

4. *Defuzzification Layer:* The normalized rule outputs are aggregated to produce crisp predictions.

Definition 4.8. For a multi-layer P-GNN, the l-th layer is defined similarly, incorporating plithogenic operations in $f_\theta^{(l)}$.

Theorem 4.9. *The Plithogenic Graph Neural Network (P-GNN) generalizes both the Neutrosophic Graph Neural Network (N-GNN) and the Fuzzy Graph Neural Network (F-GNN).*

Proof. In a P-GNN, each vertex and edge is associated with degrees of appurtenance and contradiction. Consider the special case where the degrees of contradiction are zero for all vertices and edges, and the plithogenic membership reduces to neutrosophic membership with degrees of truth, indeterminacy, and falsity. Under this condition, the P-GNN reduces to an N-GNN.

Further, if we also set the indeterminacy and falsity components to zero, the neutrosophic membership reduces to fuzzy membership, and the P-GNN reduces to an F-GNN.

Therefore, the P-GNN generalizes both the N-GNN and the F-GNN. □

Corollary 4.10. *The Plithogenic Graph Neural Network can generalize the Hesitant Fuzzy Graph Neural Network [162].*

Proof. A Hesitant Fuzzy Set [375, 376] can be generalized by a Plithogenic Set. Similarly, a Hesitant Fuzzy Graph can be generalized by a Plithogenic Graph. Therefore, following the same reasoning as for Neutrosophic Graphs, the Plithogenic Graph Neural Network generalizes the Hesitant Fuzzy Graph Neural Network. □

Theorem 4.11. *A Plithogenic Graph Neural Network (P-GNN), as defined, possesses the structural properties of a Plithogenic Graph.*

Proof. In a Plithogenic Graph $PG = (PM, PN)$, each vertex $v \in M$ is associated with:

- An attribute l and a set of possible values Ml.
- A Degree of Appurtenance Function (DAF) $adf : M \times Ml \to [0,1]^s$.
- A Degree of Contradiction Function (DCF) $aCf : Ml \times Ml \to [0,1]^t$.

Similarly, each edge $e \in N$ is associated with:

- An attribute m and a set of possible values Nm.
- A DAF $bdf : N \times Nm \to [0,1]^s$.
- A DCF $bCf : Nm \times Nm \to [0,1]^t$.

The plithogenic fuzzification functions $\mathcal{P}_V$ and $\mathcal{P}_E$ in the P-GNN assign these plithogenic memberships, satisfying the structural requirements.

In a Plithogenic Graph, for all $(x,a),(y,b) \in M \times Ml$,

$$bdf\,((xy),(a,b)) \leq \min\{adf(x,a), adf(y,b)\}.$$

In the rule layer $\mathcal{R}_P$ of the P-GNN, edge DAFs are computed based on vertex DAFs using logical rules, ensuring this constraint.

Plithogenic graphs impose reflexivity and symmetry constraints:

$$\begin{aligned} aCf(a,a) &= 0, & \forall a \in Ml,\\ aCf(a,b) &= aCf(b,a), & \forall a,b \in Ml,\\ bCf(m,m) &= 0, & \forall m \in Nm,\\ bCf(m,n) &= bCf(n,m), & \forall m,n \in Nm. \end{aligned}$$

The P-GNN enforces these constraints through its contradiction functions aCf and bCf, ensuring compliance.

The P-GNN propagates plithogenic properties through the rule layer $\mathcal{R}_P$ and defuzzification layer $\mathcal{D}_P$, maintaining structural consistency.

The P-GNN satisfies all the defining properties of a Plithogenic Graph, thus proving the theorem. □

Theorem 4.12. *In a P-GNN, the degrees of appurtenance and contradiction are preserved during the aggregation process across the network layers.*

Proof. The plithogenic aggregation functions in the P-GNN operate as follows:

1. At layer l, the updated DAF for vertex v is computed as:

$$adf^{(l)}(v, l_v) = \mathcal{A}_P \left(\{adf^{(l-1)}(u, l_u) \mid u \in \text{neighbors}(v)\}, \{bdf^{(l-1)}(e, m_e) \mid e = (v, u)\} \right),$$

   where $\mathcal{A}_P$ is the plithogenic aggregation function.

2. The updated DCFs are computed analogously, ensuring contradiction information is preserved.

As $\mathcal{A}_P$ is closed under plithogenic operations, the degrees of appurtenance and contradiction remain valid. Hence, the theorem is proven. □

Theorem 4.13. *The P-GNN can model higher levels of uncertainty and contradiction compared to traditional Graph Neural Networks (GNNs).*

Proof. The P-GNN incorporates degrees of contradiction through the DCF, which traditional GNNs do not explicitly model. Plithogenic logic extends beyond fuzzy and neutrosophic logic by introducing contradiction degrees, enabling superior expressiveness.

Thus, the P-GNN's ability to handle contradiction degrees allows it to model complex data with inherent uncertainty and contradictions, thus proving the theorem. □

Theorem 4.14. *Under certain conditions, the P-GNN converges to a stable solution that reflects the underlying plithogenic graph structure.*

Proof. The iterative updates in the P-GNN maintain the plithogenic constraints, ensuring boundedness and stability. The use of contraction mappings in the aggregation functions ensures convergence to a fixed point under suitable conditions. Thus, the P-GNN converges to a stable state that preserves the plithogenic properties, confirming the theorem. □

The algorithm for the Plithogenic Graph Neural Network is described below. We also analyze its validity, time complexity, and other relevant aspects.

Algorithm 5: Plithogenic Graph Neural Network (P-GNN)

Input: Graph $G = (V, E)$; Vertex features X_V; Edge features X_E; Number of layers L
Output: Predictions Y

foreach *vertex* $v \in V$ **do**
  Compute degrees of appurtenance and contradiction for v:
  $\alpha_v \leftarrow \text{DAF}(v)$
  $\delta_v \leftarrow \text{DCF}(v)$
end
foreach *edge* $e = (u, v) \in E$ **do**
  Compute degrees of appurtenance and contradiction for e:
  $\alpha_e \leftarrow \text{DAF}(e)$
  $\delta_e \leftarrow \text{DCF}(e)$
end
Initialize vertex representations:
$H_v^{(0)} \leftarrow X_V(v), \quad \forall v \in V$
for $l \leftarrow 1$ **to** L **do**
  foreach *vertex* $v \in V$ **do**
    Aggregate messages from neighbors:
    $m_v^{(l)} \leftarrow \sum_{u \in \mathcal{N}(v)} \gamma_{uv} \cdot H_u^{(l-1)}$
    Update vertex representation:
    $H_v^{(l)} \leftarrow \sigma\left(f_\theta^{(l)}\left(H_v^{(l-1)}, m_v^{(l)}\right)\right)$
  end
end
Compute final predictions:
$Y_v \leftarrow \text{Softmax}\left(H_v^{(L)}\right), \quad \forall v \in V$

Remark 4.15 (Algorithm Explanation)**.** *A brief description of the algorithm is provided below.*

- *Input: The algorithm takes as input a graph $G = (V, E)$, vertex features X_V, edge features X_E, and the number of layers L.*
- *Degrees of Appurtenance and Contradiction: For each vertex and edge, compute the Degree of Appurtenance Function (DAF) and Degree of Contradiction Function (DCF) as defined in the plithogenic framework.*
- *Message Passing: For each vertex v, aggregate messages from its neighbors $\mathcal{N}(v)$, weighted by a coefficient γ_{uv} that incorporates the degrees of appurtenance and contradiction:*

$$\gamma_{uv} = \mathit{Comb}\left(\alpha_u, \delta_{uv}\right),$$

  where $\mathit{Comb}(\cdot)$ is a combination function suitable for plithogenic logic.
- *Update Rule: Update the vertex representations using a trainable function $f_\theta^{(l)}$ and an activation function σ (e.g., ReLU).*
- *Output: After L layers, compute the final predictions using the Softmax function.*

Theorem 4.16 (Algorithm Validity)**.** *The P-GNN algorithm correctly computes the predictions Y according to the plithogenic logic framework.*

Proof. The P-GNN algorithm integrates plithogenic logic into the message-passing framework of graph neural networks. By computing the degrees of appurtenance (α_v, α_e) and contradiction (δ_v, δ_e) for each vertex and edge, the algorithm captures the plithogenic properties of the graph.

During message passing, the aggregation coefficient γ_{uv} combines the appurtenance and contradiction degrees using a suitable combination function. This ensures that messages are weighted appropriately based on the plithogenic relationships between vertices.

The update rule incorporates the aggregated messages and the previous vertex representation, allowing the model to learn complex patterns in the data. The use of activation functions and trainable parameters ensures that the model can approximate any continuous function, according to the universal approximation theorem.

Therefore, the algorithm correctly implements the plithogenic logic within the graph neural network framework, leading to accurate predictions Y. □

Theorem 4.17 (Time Complexity)**.** *The time complexity of the P-GNN algorithm is $O(L \cdot (|V|d + |E|d))$, where $|V|$ is the number of vertices, $|E|$ is the number of edges, and d is the dimensionality of the feature vectors.*

Proof. The time complexity analysis is as follows:

- *Degrees Computation:*
  - For vertices: Computing α_v and δ_v for all $v \in V$ takes $O(|V|)$ time.
  - For edges: Computing α_e and δ_e for all $e \in E$ takes $O(|E|)$ time.
- *Initialization:* Initializing $H_v^{(0)}$ for all $v \in V$ takes $O(|V|d)$ time.
- *Message Passing and Update (per layer):*
  - Aggregation: For each vertex $v \in V$, aggregating messages from neighbors involves:

$$m_v^{(l)} = \sum_{u \in \mathcal{N}(v)} \gamma_{uv} \cdot H_u^{(l-1)}$$

    Assuming the average degree is $\bar{k}$, this takes $O(\bar{k}d)$ time per vertex, totaling $O(|V|\bar{k}d)$ per layer.
  - Update: Updating $H_v^{(l)}$ for all $v \in V$ takes $O(|V|d)$ time per layer.
- *Total per Layer:* $O(|V|\bar{k}d)$ (since $\bar{k}$ is constant for sparse graphs, this simplifies to $O(|V|d)$).
- *Total for L Layers:* $O(L \cdot |V|d)$
- *Overall Time Complexity:* Including the degrees computation and message passing over L layers:

$$O(|V| + |E| + L \cdot |V|d) = O(L \cdot |V|d + |E|)$$

  For graphs where $|E|$ is $O(|V|)$ (sparse graphs), the complexity simplifies to $O(L \cdot |V|d)$.

□

Theorem 4.18 (Space Complexity)**.** *The space complexity of the P-GNN algorithm is $O(|V|d + |E|)$.*

Proof. The space complexity analysis is as follows:

- *Vertex Representations:* Storing $H_v^{(l)}$ for all $v \in V$ and all $l = 0, \ldots, L$ requires $O(L \cdot |V|d)$ space. However, if we overwrite $H_v^{(l-1)}$ with $H_v^{(l)}$ at each layer (i.e., do not store all previous layers), the space required reduces to $O(|V|d)$.
- *Degrees of Appurtenance and Contradiction:* Storing α_v, δ_v for all $v \in V$ requires $O(|V|)$ space. Similarly, storing α_e, δ_e for all $e \in E$ requires $O(|E|)$ space.
- *Aggregation Messages:* Storing $m_v^{(l)}$ for all $v \in V$ requires $O(|V|d)$ space.
- *Total Space Complexity:* Combining the above, the total space complexity is:

$$O(|V|d + |E| + |V|) = O(|V|d + |E|)$$

  Since $|V|d$ generally dominates $|V|$, and for sparse graphs $|E|$ is $O(|V|)$, the overall space complexity remains $O(|V|d)$.

□

4.3 Fuzzy Hypergraph Neural Network

The concept of a Fuzzy Hypergraph Neural Network integrates the principles of Hypergraph Neural Networks and Fuzzy Neural Networks. It can also be understood as a neural network representation of a Fuzzy Hypergraph. Similar to Fuzzy Graphs, extensive research has been conducted on Fuzzy Hypergraphs [11, 16, 52, 59, 98, 99, 284, 285, 396]. The relevant definitions and theorems are presented below.

Definition 4.19 (Fuzzy Hypergraph)**.** [311] Let X be a finite set of vertices, and let E be a finite family of non-trivial fuzzy subsets of X, where each fuzzy set $A \in E$ is defined by a membership function $\mu_A : X \to [0, 1]$. A pair $H = (X, E)$ is called a *Fuzzy Hypergraph* if the following conditions are satisfied:

- $X = \bigcup\{\text{supp}(A) \mid A \in E\}$, where the *support* of a fuzzy set A is defined as $\text{supp}(A) = \{x \in X \mid \mu_A(x) > 0\}$.
- E is the *fuzzy edge set*, consisting of fuzzy subsets of X.

The *height* of a fuzzy hypergraph H, denoted $h(H)$, is defined as:

$$h(H) = \max\{\max_{x \in X} \mu_A(x) \mid A \in E\}.$$

A Fuzzy Hypergraph $H = (X, E)$ is:

- *Simple* if E contains no repeated fuzzy edges and, for any $A, B \in E$ with $A \subseteq B$, it follows that $A = B$.
- *Support Simple* if $A, B \in E$, $A \subseteq B$, and $\text{supp}(A) = \text{supp}(B)$, then $A = B$.

Definition 4.20 (Crisp Level Hypergraph of a Fuzzy Hypergraph)**.** Let $H = (X, E)$ be a Fuzzy Hypergraph. For a threshold $c \in (0, 1]$, the c-cut (or c-level) of a fuzzy edge $A \in E$ is defined as:

$$A_c = \{x \in X \mid \mu_A(x) \geq c\}.$$

The c-level hypergraph $H_c = (X_c, E_c)$ of H is defined as:

$$X_c = \bigcup\{A_c \mid A \in E\}, \quad E_c = \{A_c \mid A \in E\}.$$

Theorem 4.21. *(cf.[15, 268]) A Fuzzy Hypergraph generalizes both Fuzzy Graphs and (crisp) Hypergraphs.*

Proof. A Fuzzy Graph $G = (X, E, \mu_V, \mu_E)$ is a special case of a Fuzzy Hypergraph $H = (X, E)$, where:

- The vertex membership function $\mu_V : X \to [0, 1]$ in G corresponds to the vertex set X in H.
- Each edge membership function $\mu_E : X \times X \to [0, 1]$ in G can be represented as a fuzzy subset $A \in E$ in H, where $A \subseteq X$ and $\mu_A(x) = \max\{\mu_E(x, y) \mid y \in X\}$.

Thus, a Fuzzy Graph is a Fuzzy Hypergraph where each edge connects at most two vertices.
A Hypergraph $H^* = (X, E)$ is a special case of a Fuzzy Hypergraph $H = (X, E)$, where:

- Each edge $A \in E$ in H^* is a crisp subset of X, corresponding to a fuzzy edge in H with $\mu_A(x) \in \{0, 1\}$ for all $x \in X$.
- The membership function of each fuzzy edge A in H reduces to an indicator function, $\mu_A(x) = 1$ if $x \in A$, and $\mu_A(x) = 0$ otherwise.

Hence, a Hypergraph is a Fuzzy Hypergraph where all edges are crisp subsets. □

Definition 4.22 (Fuzzy incidence matrix)**.** The *fuzzy incidence matrix* $H_f \in \mathbb{R}^{n \times m}$ of the fuzzy hypergraph H is defined by:

$$(H_f)_{ij} = \mu_{A_j}(x_i),$$

where $x_i \in X$ and $A_j \in E$.

The *fuzzy degree* of a vertex $x_i \in X$ is defined as:

$$d(x_i) = \sum_{j=1}^{m} (H_f)_{ij} w_j,$$

where w_j is the weight of fuzzy hyperedge A_j.

The *fuzzy degree* of a hyperedge $A_j \in E$ is defined as:

$$\delta(A_j) = \sum_{i=1}^{n} (H_f)_{ij}.$$

Let $D_V \in \mathbb{R}^{n\times n}$ and $D_E \in \mathbb{R}^{m\times m}$ be the diagonal matrices of fuzzy vertex degrees and fuzzy hyperedge degrees, respectively:

$$(D_V)_{ii} = d(x_i), \quad (D_E)_{jj} = \delta(A_j).$$

Theorem 4.23. *The fuzzy incidence matrix H_f can represent both a Fuzzy Hypergraph and a Hypergraph as special cases.*

Proof. Let $H = (X, E)$ be a Fuzzy Hypergraph, where $X = \{x_1, x_2, \ldots, x_n\}$ is the set of vertices and $E = \{A_1, A_2, \ldots, A_m\}$ is the fuzzy edge set. Each fuzzy edge A_j is defined by a membership function $\mu_{A_j} : X \to [0, 1]$. The fuzzy incidence matrix $H_f \in \mathbb{R}^{n\times m}$ is defined as:

$$(H_f)_{ij} = \mu_{A_j}(x_i),$$

where $\mu_{A_j}(x_i) \in [0, 1]$ represents the degree of membership of vertex x_i in the fuzzy edge A_j.

The rows of H_f correspond to the vertices $x_i \in X$, and the columns correspond to the fuzzy edges $A_j \in E$. The support of each fuzzy edge A_j can be recovered as:

$$\text{supp}(A_j) = \{x_i \in X \mid (H_f)_{ij} > 0\}.$$

The vertex degrees $d(x_i)$ and hyperedge degrees $\delta(A_j)$ are defined in terms of H_f, as shown in the definition of the fuzzy incidence matrix. Thus, H_f fully encodes the structure of the Fuzzy Hypergraph.

A Hypergraph $\mathcal{H} = (X, E)$ is a special case of a Fuzzy Hypergraph where all membership values are binary, i.e., $\mu_{A_j}(x_i) \in \{0, 1\}$. In this case, the incidence matrix H_f reduces to the classical incidence matrix H, where:

$$(H)_{ij} = \begin{cases} 1, & \text{if } x_i \in A_j, \\ 0, & \text{otherwise.} \end{cases}$$

For binary $\mu_{A_j}(x_i)$, the support of each edge A_j is:

$$\text{supp}(A_j) = \{x_i \in X \mid \mu_{A_j}(x_i) = 1\},$$

which matches the standard definition of a hyperedge in a Hypergraph. The vertex and hyperedge degree definitions also simplify to their classical counterparts:

$$d(x_i) = \sum_{j=1}^{m} (H)_{ij}, \quad \delta(A_j) = \sum_{i=1}^{n} (H)_{ij}.$$

The fuzzy incidence matrix H_f generalizes the classical incidence matrix H, allowing it to represent both Fuzzy Hypergraphs and Hypergraphs. By setting $\mu_{A_j}(x_i) \in [0, 1]$, it represents a Fuzzy Hypergraph, and by restricting $\mu_{A_j}(x_i)$ to binary values, it represents a Hypergraph. □

Definition 4.24 (Fuzzy Hypergraph Laplacian)**.** The *fuzzy hypergraph Laplacian* Δ_f is defined as:

$$\Delta_f = I - D_V^{-1/2} H_f W D_E^{-1} H_f^\top D_V^{-1/2},$$

where $W = \text{diag}(w_1, w_2, \ldots, w_m)$ is the diagonal matrix of fuzzy hyperedge weights, and I is the identity matrix.

Theorem 4.25. *The Fuzzy Hypergraph Laplacian Δ_f generalizes the Hypergraph Laplacian L.*

Proof. 1. *Generalization Setup:*
The fuzzy hypergraph Laplacian Δ_f is defined as:

$$\Delta_f = I - D_V^{-1/2} H_f W D_E^{-1} H_f^\top D_V^{-1/2},$$

where H_f is the fuzzy incidence matrix, and W is the diagonal matrix of fuzzy hyperedge weights. The hypergraph Laplacian L is a special case of this construction, defined as:

$$L = I - D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2}.$$

2. *Connection Between H and H_f:*
The classical incidence matrix H is binary, with entries:

$$H_{ij} = \begin{cases} 1, & \text{if } v_i \in e_j, \\ 0, & \text{otherwise.} \end{cases}$$

In contrast, the fuzzy incidence matrix H_f allows entries $H_{ij}^f \in [0, 1]$, representing the degree of membership of vertex v_i in hyperedge e_j. When H_f is restricted to binary values, it coincides with H.

3. *Generalization of Matrices:*

- *Vertex Degree Matrix:* In the classical case, the diagonal vertex degree matrix D_v has entries:

$$(D_v)_{ii} = \sum_{j=1}^{m} H_{ij} w(e_j).$$

  In the fuzzy case, this generalizes to:

$$(D_V)_{ii} = \sum_{j=1}^{m} H_{ij}^f w(e_j),$$

  allowing H_{ij}^f to take non-binary values.

- *Hyperedge Degree Matrix:* Similarly, the hyperedge degree matrix D_e generalizes to:

$$(D_E)_{jj} = \sum_{i=1}^{n} H_{ij}^f.$$

4. *Substitution in Δ_f:*
Substituting the generalized H_f, D_V, and D_E into Δ_f, we recover the classical Laplacian L when H_f is binary. This shows that L is a special case of Δ_f.

Since Δ_f reduces to L under binary constraints on H_f and the associated matrices, Δ_f is a generalization of L.

Thus, the Fuzzy Hypergraph Laplacian generalizes the Hypergraph Laplacian by extending the binary incidence matrix to a fuzzy membership matrix, enabling the representation of partial or uncertain membership relationships. □

Definition 4.26 (Fuzzy Hypergraph Neural Network)**.** An *Fuzzy Hypergraph Neural Network* (F-HGNN) is a neural network designed to operate on fuzzy hypergraphs. Given a fuzzy hypergraph $H = (X, E)$ with fuzzy incidence matrix H_f, vertex feature matrix $X \in \mathbb{R}^{n \times d}$, and fuzzy hyperedge weight matrix W, the F-HGNN performs convolution operations defined as:

$$Y = \sigma\left(D_V^{-1/2} H_f W D_E^{-1} H_f^\top D_V^{-1/2} X \Theta\right),$$

where:

- σ is an activation function (e.g., ReLU).
- $\Theta \in \mathbb{R}^{d \times c}$ is the learnable weight matrix.
- $Y \in \mathbb{R}^{n \times c}$ is the output feature matrix.

Definition 4.27 (Multi-Layer F-HGNN)**.** For a multi-layer F-HGNN, the l-th layer's output is computed as:

$$X^{(l+1)} = \sigma\left(D_V^{-1/2} H_f W D_E^{-1} H_f^\top D_V^{-1/2} X^{(l)} \Theta^{(l)}\right),$$

where $X^{(0)}$ is the input feature matrix, and $\Theta^{(l)}$ is the learnable weight matrix at layer l.

Theorem 4.28. *The Fuzzy Hypergraph Neural Network (F-HGNN) generalizes both the Hypergraph Neural Network (HGNN) and the Fuzzy Graph Neural Network (F-GNN).*

Proof. We will prove that:

1. When the fuzzy hypergraph reduces to a crisp hypergraph (i.e., membership functions $\mu_A(x) \in \{0, 1\}$), the F-HGNN reduces to the HGNN.
2. When the hyperedges are fuzzy edges connecting at most two vertices, the F-HGNN reduces to the F-GNN.

Case 1: F-HGNN Reduces to HGNN

Assume that the fuzzy hypergraph $H = (X, E)$ is crisp; that is, for all $A \in E$ and $x \in X$, the membership functions $\mu_A(x) \in \{0, 1\}$.

In this case, the fuzzy incidence matrix H_f becomes the standard incidence matrix H of a hypergraph, where:

$$(H_f)_{ij} = \mu_{A_j}(x_i) = \begin{cases} 1, & \text{if } x_i \in A_j, \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, the fuzzy vertex degrees $d(x_i)$ and hyperedge degrees $\delta(A_j)$ become the standard degrees in a hypergraph.

Therefore, the F-HGNN convolution operation simplifies to:

$$Y = \sigma\left(D_V^{-1/2} H W D_E^{-1} H^\top D_V^{-1/2} X \Theta\right),$$

which is exactly the convolution operation in the Hypergraph Neural Network (HGNN).

Case 2: F-HGNN Reduces to F-GNN

Assume that each fuzzy hyperedge $A_j \in E$ connects at most two vertices. This means that the supports of A_j are such that $|\operatorname{supp}(A_j)| \leq 2$.

In this case, the fuzzy hypergraph reduces to a fuzzy graph, where edges are fuzzy and connect two vertices. The fuzzy incidence matrix H_f becomes analogous to the adjacency representation in a fuzzy graph.

The convolution operation in F-HGNN becomes similar to that in Fuzzy Graph Neural Networks, where messages are passed between connected vertices, weighted by the fuzzy membership degrees.

Therefore, the F-HGNN generalizes the F-GNN in this case.

Since F-HGNN reduces to HGNN when the fuzzy hypergraph is crisp, and reduces to F-GNN when hyperedges connect at most two vertices, we conclude that F-HGNN generalizes both HGNN and F-GNN. □

Theorem 4.29. *A Fuzzy Hypergraph Neural Network (F-HGNN) retains the structure of a Fuzzy Hypergraph.*

Proof. The Fuzzy Hypergraph Neural Network (F-HGNN) operates on the fuzzy incidence matrix H_f of a Fuzzy Hypergraph $H = (X, E)$. All transformations, including convolution operations, rely on H_f, which encodes the fuzzy edge membership functions $\mu_A(x)$ of $A \in E$.

Since the operations preserve the relationships defined by H_f, the structure of the Fuzzy Hypergraph H is inherently retained throughout the F-HGNN's computations. □

Question 4.30. Is it possible to extend the concept by utilizing Neutrosophic Hypergraphs [13, 14, 19, 248, 249, 255] and Plithogenic Hypergraphs [258]?

5 Other SuperHyperGraph Concepts

In this section, we explore concepts related to SuperHyperGraphs that are not directly connected to the topics discussed above.

5.1 Multilevel k-way Hypergraph Partitioning

Multilevel graph partitioning is an approach to divide a graph into smaller parts by iteratively coarsening, partitioning, and refining it for optimization [81, 147, 216, 217]. In Hypergraph Theory, concepts such as Multilevel Hypergraph Partitioning [214, 215] and Multilevel k-way Hypergraph Partitioning[35, 218, 305, 317, 379] are frequently studied. These concepts are well-known for their applications in fields like VLSI design. This section considers the definition of Multilevel k-way n-SuperHyperGraph Partitioning.

Definition 5.1 (Multilevel k-way Hypergraph Partitioning)**.** [218] Given a hypergraph $H = (V, E)$, where V is the set of vertices and E is the set of hyperedges, and a positive integer k, the goal of multilevel k-way hypergraph partitioning is to partition the vertex set V into k disjoint subsets $\{V_1, V_2, \ldots, V_k\}$, such that:

1. The size of each subset satisfies the balancing constraint:

$$\frac{|V|}{k \cdot c} \leq |V_i| \leq c \cdot \frac{|V|}{k}, \quad \forall i \in \{1, 2, \ldots, k\},$$

   where $c \geq 1$ is the imbalance tolerance factor.

2. An objective function f defined over the hyperedges E is optimized. Common objectives include:

   - Minimizing the hyperedge cut:

$$f_{\text{cut}} = \sum_{e \in E} (\text{spanned_partitions}(e) - 1),$$

     where spanned_partitions(e) is the number of subsets V_i spanned by the hyperedge e.
   - Minimizing the sum of external degrees (SOED):

$$f_{\text{SOED}} = \sum_{e \in E} \text{external_degree}(e),$$

     where external_degree(e) is the number of subsets V_i that the hyperedge e spans.

   The multilevel k-way partitioning algorithm consists of three phases:

- *Coarsening Phase:* The hypergraph H is iteratively coarsened into a series of smaller hypergraphs

$$H_1, H_2, \ldots, H_\ell$$

  by merging vertices to reduce complexity.

- *Initial Partitioning Phase:* The smallest hypergraph H_ℓ is directly partitioned into k subsets using an efficient partitioning algorithm.

- *Uncoarsening Phase:* The partitioning is progressively refined as it is projected back to the original hypergraph H, using refinement algorithms such as FM or greedy approaches to optimize the objective function while maintaining the balancing constraint.

Definition 5.2 (Multilevel k-way n-SuperHyperGraph Partitioning)**.** Given an n-SuperHyperGraph $H = (V, E)$, where V is the set of supervertices and E is the set of superedges, and a positive integer k, the goal of multilevel k-way n-SuperHyperGraph Partitioning is to partition the supervertex set V into k disjoint subsets $\{V_1, V_2, \ldots, V_k\}$, such that:

1. The size of each subset satisfies the balancing constraint:

$$\frac{|V|}{k \cdot c} \leq |V_i| \leq c \cdot \frac{|V|}{k}, \quad \forall i \in \{1, 2, \ldots, k\},$$

   where $c \geq 1$ is the imbalance tolerance factor.

2. An objective function f defined over the superedges E is optimized. Common objectives include:

- *Minimizing the superedge cut*:

$$f_{\text{cut}} = \sum_{e \in E} (\text{spanned_partitions}(e) - 1),$$

where spanned_partitions(e) is the number of subsets V_i spanned by the superedge e.

- *Minimizing the sum of external degrees (SOED)*:

$$f_{\text{SOED}} = \sum_{e \in E} \text{external_degree}(e),$$

where external_degree(e) is the number of subsets V_i that the superedge e spans.

The multilevel k-way partitioning algorithm consists of three phases:

- *Coarsening Phase:* The n-SuperHyperGraph H is iteratively coarsened into a series of smaller n-SuperHyperGraphs

$$H_1, H_2, \ldots, H_\ell$$

by merging supervertices to reduce complexity.

- *Initial Partitioning Phase:* The smallest n-SuperHyperGraph H_ℓ is directly partitioned into k subsets using an efficient partitioning algorithm.
- *Uncoarsening Phase:* The partitioning is progressively refined as it is projected back to the original n-SuperHyperGraph H, using refinement algorithms to optimize the objective function while maintaining the balancing constraint.

Theorem 5.3. *The Multilevel k-way n-SuperHyperGraph Partitioning generalizes the Multilevel k-way Hypergraph Partitioning. Specifically, when $n = 1$, the Multilevel k-way n-SuperHyperGraph Partitioning reduces to the standard Multilevel k-way Hypergraph Partitioning.*

Proof. To prove that the Multilevel k-way n-SuperHyperGraph Partitioning generalizes the Multilevel k-way Hypergraph Partitioning, we need to show that when $n = 1$, the definitions coincide.

1. *At $n = 1$, the n-SuperHyperGraph reduces to a Hypergraph:*

- The 1-th iterated power set of V_0 is $\mathcal{P}^1(V_0) = \mathcal{P}(V_0)$, the power set of V_0.
- However, in standard hypergraphs, the vertex set is $V = V_0$, not $V \subseteq \mathcal{P}(V_0)$. To align the definitions, we consider only the elements of $\mathcal{P}^1(V_0)$ that are singletons. That is, $V = V_0 \subseteq \mathcal{P}(V_0)$.
- The hyperedges $E \subseteq \mathcal{P}(V_0)$, which matches the definition of hyperedges in a standard hypergraph.

2. *Partitioning Definitions Align:*

- The partitioning of supervertices V into k subsets $\{V_1, V_2, \ldots, V_k\}$ in the n-SuperHyperGraph becomes the partitioning of vertices V_0 when $n = 1$.
- The balancing constraints and objective functions remain the same, as they are defined over V and E, which now correspond to V_0 and E of the hypergraph.

3. *Algorithm Phases Correspond:*

- *Coarsening Phase:* Merging supervertices in the n-SuperHyperGraph corresponds to merging vertices in the hypergraph.
- *Initial Partitioning Phase:* Partitioning the smallest n-SuperHyperGraph aligns with partitioning the coarsest hypergraph.
- *Uncoarsening Phase:* Refinement steps are analogous in both cases.

Therefore, when $n = 1$, the Multilevel k-way n-SuperHyperGraph Partitioning reduces to the Multilevel k-way Hypergraph Partitioning, proving that the former generalizes the latter. □

5.2 Superhypergraph Random Walk

A Graph Random Walk is a discrete-time Markov chain where transitions between vertices follow edge-based probabilities, modeling stochastic processes on graphs [83, 408]. These concepts have been extended to hypergraphs, leading to the development of Hypergraph Random Walks[74, 82, 105, 174, 275]. In this subsection, we extend Hypergraph Random Walks to the domain of Superhypergraphs. The related definitions and theorems are provided below.

Definition 5.4 (Markov Chain)**.** (cf.[30, 84, 157]) A *Markov Chain* is a mathematical framework used to model stochastic processes where the future state depends solely on the current state and not on how it was reached. Formally:

- *State Space*: The set of possible states is denoted by $S = \{s_1, s_2, \dots\}$, which may be finite or countable.
- *Transition Rule*: The process satisfies the property:

$$P(X_{t+1} = s_j \mid X_t = s_i, X_{t-1}, \dots, X_0) = P(X_{t+1} = s_j \mid X_t = s_i).$$

- *Transition Matrix*: Probabilities of moving between states are organized in a matrix $P = [p_{ij}]$, with:

$$p_{ij} = P(X_{t+1} = s_j \mid X_t = s_i), \quad \text{and} \sum_{j \in S} p_{ij} = 1 \quad \forall i.$$

- *Initial State Distribution*: The process begins with probabilities $\pi_0(i) = P(X_0 = s_i)$.

Example 5.5 (Weather System (Markov Chain))**.** A simplified weather model predicts sunny (S) or rainy (R) conditions based on current weather:

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}.$$

If today is sunny, there is a 90% chance of sunshine tomorrow.

Definition 5.6 (Discrete-time Markov Chain)**.** (cf.[88, 330, 423]) A *Discrete-time Markov Chain (DTMC)* is a stochastic process $\{X_t\}_{t=0}^{\infty}$ defined on a discrete state space $S = \{s_1, s_2, \dots\}$, satisfying the *Markov property*, which states that the probability of transitioning to the next state depends only on the current state and not on the sequence of previous states. Formally:

$$P(X_{t+1} = s_j \mid X_t = s_i, X_{t-1} = s_k, \dots, X_0 = s_m) = P(X_{t+1} = s_j \mid X_t = s_i),$$

for all $t \geq 0$, $s_i, s_j \in S$, and any sequence of states $s_m, \dots, s_k, s_i$.
The dynamics of a DTMC are governed by a *transition probability matrix* $P = [p_{ij}]$, where

$$p_{ij} = P(X_{t+1} = s_j \mid X_t = s_i),$$

and

$$\sum_{j \in S} p_{ij} = 1 \quad \text{for all } i \in S.$$

The initial distribution over the states is specified by a vector π_0, where $\pi_0(i) = P(X_0 = s_i)$.

Definition 5.7 (Hypergraph Random Walk)**.** [73, 174] A *Hypergraph Random Walk* is a discrete-time Markov process defined over the vertices of a hypergraph $H = (V, E)$, with transition probabilities determined as follows:

1. *Hyperedge Selection*: Starting from the current vertex $v_t \in V$ at time t, a hyperedge $e \in E$ containing v_t is selected with probability proportional to its weight $\omega(e) > 0$. Formally, the selection probability is:

$$P(e \mid v_t) = \frac{\omega(e)}{\sum_{e' \ni v_t} \omega(e')}.$$

2. *Vertex Selection within the Hyperedge*: From the selected hyperedge e, a vertex $v_{t+1} \in e$ is chosen. This selection can follow either:

a) *Uniform Selection*: Choose v_{t+1} uniformly at random from e, such that:

$$P(v_{t+1} \mid e) = \frac{1}{|e|}.$$

b) *Weighted Selection*: Choose v_{t+1} based on a vertex-specific weight $\gamma_e(v) > 0$ within e, such that:

$$P(v_{t+1} \mid e) = \frac{\gamma_e(v_{t+1})}{\sum_{v \in e} \gamma_e(v)}.$$

The full transition probability from v_t to v_{t+1} is then given by:

$$P(v_{t+1} \mid v_t) = \sum_{e \ni v_t, v_{t+1}} P(e \mid v_t) \cdot P(v_{t+1} \mid e).$$

This formulation generalizes random walks on graphs by accounting for hyperedges that can connect more than two vertices.

Definition 5.8 (n-SuperHyperGraph Random Walk)**.** Let $H = (V, E)$ be an n-SuperHyperGraph, where $V \subseteq \mathcal{P}^n(V_0)$ is the set of supervertices, and $E \subseteq \mathcal{P}^n(V_0)$ is the set of superedges. Here, $\mathcal{P}^n(V_0)$ denotes the n-th iterated power set of the base set V_0.

A *n-SuperHyperGraph Random Walk* is a discrete-time stochastic process $\{X_t\}_{t=0}^{\infty}$ defined on the supervertices V, with transitions determined as follows:

1. *Superedge Selection*: Starting from the current supervertex $v_t \in V$ at time t, select a superedge $e \in E$ containing v_t, with probability proportional to its weight $\omega(e) > 0$:

$$P(e \mid v_t) = \frac{\omega(e)}{\sum_{e' \ni v_t} \omega(e')}.$$

2. *Supervertex Selection within the Superedge*: From the selected superedge e, select a supervertex $v_{t+1} \in e$ according to a probability distribution, which can be:

   a) *Uniform Selection*: Choose v_{t+1} uniformly at random from e:

$$P(v_{t+1} \mid e) = \frac{1}{|e|}.$$

   b) *Weighted Selection*: Choose v_{t+1} based on weights $\gamma_e(v) > 0$:

$$P(v_{t+1} \mid e) = \frac{\gamma_e(v_{t+1})}{\sum_{v \in e} \gamma_e(v)}.$$

   The full transition probability from v_t to v_{t+1} is then:

$$P(v_{t+1} \mid v_t) = \sum_{e \ni v_t, v_{t+1}} P(e \mid v_t) \cdot P(v_{t+1} \mid e).$$

Theorem 5.9. *The n-SuperHyperGraph Random Walk has the structure of an n-SuperHyperGraph.*

Proof. Since the random walk is defined over supervertices $V \subseteq \mathcal{P}^n(V_0)$ and utilizes superedges $E \subseteq \mathcal{P}^n(V_0)$ for transitions, it inherently possesses the structure of an n-SuperHyperGraph. □

Corollary 5.10. *The n-SuperHyperGraph Random Walk possesses the structure of a superhypergraph, hypergraph, and graph.*

Proof. This follows directly from the above theorem. □

Theorem 5.11. *The n-SuperHyperGraph Random Walk is a Discrete-time Markov Chain.*

Proof. The process $\{X_t\}$ satisfies the Markov property because the probability of transitioning to v_{t+1} depends only on the current supervertex v_t and not on any previous supervertices $v_{t-1}, v_{t-2}, \ldots$. The transition probabilities $P(v_{t+1} \mid v_t)$ are well-defined, and the process evolves in discrete time steps. Therefore, it is a Discrete-time Markov Chain. □

Theorem 5.12. *The n-SuperHyperGraph Random Walk generalizes the Hypergraph Random Walk.*

Proof. When $n = 1$, the n-SuperHyperGraph reduces to a standard hypergraph, and the n-SuperHyperGraph Random Walk becomes equivalent to the Hypergraph Random Walk. Therefore, the n-SuperHyperGraph Random Walk is a generalization of the Hypergraph Random Walk. □

Question 5.13. The concept of HyperRandom [149–151], which extends the idea of randomness, is well-known. Can this be used to further extend the concept of Random Walk?

5.3 Superhypergraph Turán Problem

The Hypergraph Turán Problem [165, 219, 241] aims to determine the maximum number of edges in a uniform hypergraph (cf.[184, 185, 203]) on n vertices while avoiding a specific forbidden subhypergraph. This concept is extended to superhypergraphs, and their characteristics are briefly examined. The relevant definitions and theorems are presented below.

Definition 5.14 (Forbidden Graph)**.** (cf.[106]) A *forbidden graph* F is a graph that is not allowed as a subgraph in a larger graph G. If G contains F as a subgraph, G violates the specified constraints, often used in Turán-type problems or graph property investigations.

Definition 5.15 (Hypergraph Turán Problem)**.** [219] Let $G = (V, E)$ be an r-uniform hypergraph, where V is the set of vertices and E is the set of edges, with each edge being a subset of V containing exactly r vertices.

Let F be any r-uniform hypergraph. A hypergraph G is said to be F-free if G does not contain F as a subhypergraph.

The *Hypergraph Turán Number* $\mathrm{ex}_r(n, F)$ is defined as the maximum number of edges in an F-free r-uniform hypergraph on n vertices:

$$\mathrm{ex}_r(n, F) = \max\{|E(G)| : G \text{ is an } F\text{-free } r\text{-uniform hypergraph with } |V(G)| = n\}.$$

Furthermore, the *Turán Density* $\pi(F)$ of F is given by:

$$\pi(F) = \lim_{n\to\infty} \frac{\mathrm{ex}_r(n, F)}{\binom{n}{r}},$$

where $\binom{n}{r}$ denotes the number of all possible r-element subsets of n vertices.

Definition 5.16 (r-Uniform n-SuperHyperGraph)**.** An n-SuperHyperGraph $H = (V, E)$ is called *r-uniform* if every superedge $e \in E$ contains exactly r supervertices, i.e., $e \subseteq V$ and $|e| = r$.

Definition 5.17 (n-SuperHyperGraph Turán Problem)**.** Let F be an r-uniform n-SuperHyperGraph.

An r-uniform n-SuperHyperGraph $G = (V, E)$ is said to be *F-free* if G does not contain F as a subgraph.

The *n-SuperHyperGraph Turán Number* $\mathrm{ex}_r^n(N, F)$ is defined as the maximum number of edges in an F-free r-uniform n-SuperHyperGraph G with $|V(G)| = N$:

$$\mathrm{ex}_r^n(N, F) = \max\{|E(G)| : G \text{ is an } F\text{-free } r\text{-uniform } n\text{-SuperHyperGraph with } |V(G)| = N\}.$$

Furthermore, the *n-SuperHyperGraph Turán Density* $\pi^n(F)$ is defined as:

$$\pi^n(F) = \lim_{N\to\infty} \frac{\mathrm{ex}_r^n(N, F)}{\binom{N}{r}},$$

where $\binom{N}{r}$ denotes the number of all possible r-element subsets of N supervertices.

Theorem 5.18. *An r-uniform hypergraph is a special case of an r-uniform n-SuperHyperGraph when $n = 0$.*

Proof. When $n = 0$, we have $\mathcal{P}^0(V_0) = V_0$. Thus, the supervertices V are exactly the base vertices V_0. The superedges E are subsets of V containing exactly r supervertices. Therefore, an r-uniform 0-SuperHyperGraph $H = (V, E)$ is identical to an r-uniform hypergraph on the vertex set V_0. □

Theorem 5.19. *Every r-uniform hypergraph can be represented as an r-uniform n-SuperHyperGraph for any $n \geq 0$.*

Proof. Given an r-uniform hypergraph $H = (V_0, E)$, we can construct an r-uniform n-SuperHyperGraph $H' = (V, E')$ by setting $V = V_0 \subseteq \mathcal{P}^n(V_0)$ and $E' = E$. Since the supervertices V are the original vertices V_0, and the superedges E' are the same as E, H' is an r-uniform n-SuperHyperGraph equivalent to H. □

Theorem 5.20. *The n-SuperHyperGraph Turán Problem generalizes the Hypergraph Turán Problem.*

Proof. When $n = 0$, the n-SuperHyperGraph Turán Problem reduces to the classical Hypergraph Turán Problem because the supervertices are the original vertices V_0, and the superedges are subsets of V_0 of size r. Therefore, the n-SuperHyperGraph Turán Problem includes the Hypergraph Turán Problem as a special case, thus generalizing it. □

Theorem 5.21. *For any r-uniform hypergraph F, the Hypergraph Turán Number $\mathrm{ex}_r(N, F)$ is less than or equal to the n-SuperHyperGraph Turán Number $\mathrm{ex}_r^n(N, F')$, where F' is the corresponding r-uniform n-SuperHyperGraph constructed from F.*

Proof. Since every r-uniform hypergraph G can be viewed as an r-uniform n-SuperHyperGraph G' by treating vertices as supervertices (as per the previous theorem), any F-free r-uniform hypergraph G corresponds to an F'-free r-uniform n-SuperHyperGraph G'. However, the set of r-uniform n-SuperHyperGraphs includes more general structures due to the hierarchical nature of supervertices. Therefore, there may exist F'-free r-uniform n-SuperHyperGraphs with more edges than any F-free r-uniform hypergraph. Thus,

$$\mathrm{ex}_r(N, F) \leq \mathrm{ex}_r^n(N, F').$$

□

Corollary 5.22. *The Turán Density of an r-uniform hypergraph F satisfies:*

$$\pi(F) \leq \pi^n(F'),$$

where F' is the corresponding r-uniform n-SuperHyperGraph constructed from F.

Proof. This follows directly from the previous theorem and the definitions of Turán Densities:

$$\pi(F) = \lim_{N\to\infty} \frac{\mathrm{ex}_r(N, F)}{\binom{N}{r}} \leq \lim_{N\to\infty} \frac{\mathrm{ex}_r^n(N, F')}{\binom{N}{r}} = \pi^n(F').$$

□

Theorem 5.23. *An n-SuperHyperGraph Turán Number can be strictly greater than the corresponding Hypergraph Turán Number.*

Proof. Due to the additional complexity and hierarchical structure of supervertices in an n-SuperHyperGraph, there are more possibilities for constructing F-free r-uniform n-SuperHyperGraphs with more edges than possible in the standard hypergraph case. Therefore, for certain F and sufficiently large n, we have:

$$\mathrm{ex}_r(N, F) < \mathrm{ex}_r^n(N, F').$$

□

5.4 Binary decision n-superhypertree

A Binary Decision Hypertree is a rooted acyclic graph representing Boolean function evaluations, branching on variables with outputs at leaves [168, 169]. This concept is extended to the superhyper framework. The definitions and theorems are provided below.

Definition 5.24 (hyperdiagram)**.** (cf. [168])
A *hyperdiagram* on a finite set $G = \{x_1, x_2, \ldots, x_n\}$ is an ordered pair $H = (G, \{E_k\}_{k=1}^m)$ where:

- For each $1 \leq k \leq m$, $E_k \subseteq G$ and $|E_k| \geq 1$.

Definition 5.25 (n-Superhyperdiagram)**.** Let V_0 be a finite set of base elements. Define the n-th iterated power set of V_0 recursively as:

$$\mathcal{P}^0(V_0) = V_0, \quad \mathcal{P}^{k+1}(V_0) = \mathcal{P}\left(\mathcal{P}^k(V_0)\right),$$

where $\mathcal{P}(A)$ denotes the power set of set A.
An *n-Superhyperdiagram* is an ordered pair $H = (V, \{E_k\}_{k=1}^m)$ where:

- $V \subseteq \mathcal{P}^n(V_0)$ is the set of *supervertices*.
- For each $1 \leq k \leq m$, $E_k \subseteq V$ is called a *superedge* (or hyperedge), with $|E_k| \geq 1$.

Theorem 5.26. *An n-Superhyperdiagram generalizes the hyperdiagram.*

Proof. When $n = 0$, the n-th iterated power set is $\mathcal{P}^0(V_0) = V_0$. Therefore, the supervertices $V \subseteq \mathcal{P}^0(V_0) = V_0$ are simply elements of the base set V_0.
Thus, when $n = 0$, an n-Superhyperdiagram $H = (V, \{E_k\}_{k=1}^m)$ reduces to a hyperdiagram on V_0, since $V = V_0$ and each $E_k \subseteq V$.
Therefore, the concept of a hyperdiagram is a special case of an n-Superhyperdiagram when $n = 0$. Thus, n-Superhyperdiagrams generalize hyperdiagrams. □

Definition 5.27 (Binary Decision Hypertree)**.** (cf. [169])
A *Binary Decision Hypertree* is a rooted tree constructed from a Boolean function f where:

- Each node corresponds to a variable $x_i \in V_0$.
- Each internal node has two outgoing edges representing $x_i = 1$ and $x_i = 0$.
- Leaves are labeled with the output of f.

Definition 5.28 (Binary Decision n-Superhypertree)**.** Let V_0 be a finite set of variables. Consider a Boolean function f defined on V_0. A *Binary Decision n-Superhypertree* (BDnSHT) is a rooted tree constructed as follows:

- Each node represents a supervertex $v \in \mathcal{P}^n(V_0)$.
- Internal nodes are associated with testing a variable $x_i \in V_0$.
- Each internal node has two outgoing edges:
  - A solid directed edge representing the assignment $x_i = 1$.
  - A dashed directed edge representing the assignment $x_i = 0$.
- Leaves are labeled with the output value of the function f corresponding to the path from the root to the leaf.

Theorem 5.29. *A binary decision n-superhypertree generalizes the binary decision hypertree.*

Proof. When $n = 0$, the n-th iterated power set is $\mathcal{P}^0(V_0) = V_0$, so the supervertices are simply the base variables V_0.
In a binary decision hypertree, nodes correspond to variables $x_i \in V_0$, and the tree represents the evaluation of the Boolean function f by branching on the assignments of these variables.
Therefore, when $n = 0$, the binary decision n-superhypertree reduces to the binary decision hypertree.
Thus, the binary decision n-superhypertree generalizes the binary decision hypertree. □

6 Future Directions of this Research

This section highlights potential future directions for this research. A key objective is the practical implementation and experimental validation of the SuperHyperGraph Neural Network (SHGNN). Through computational experiments, we hope to discover related concepts that make the SHGNN more suitable for practical applications.

Another promising avenue is the exploration of extensions to SuperHyperGraph Neural Networks incorporating Fuzzy sets [306, 430–437] and Neutrosophic sets [126, 133, 134, 332–337, 353, 356, 359]. This includes developing and validating frameworks such as Fuzzy SuperHyperGraph Neural Networks and Neutrosophic SuperHyperGraph Neural Networks. These frameworks aim to generalize Fuzzy Neural Networks [176, 242, 250, 365, 366] and Neutrosophic Neural Networks [194] by integrating the structural advantages of hypergraphs, laying the groundwork for advanced representations and computations. Additionally, future research could explore considerations involving Directed SuperHyperGraphs and their applications [126].

In addition to the concepts mentioned above, numerous frameworks for handling uncertainty, such as Soft Set (Soft Graph) [127, 254, 266], hypersoft set[2, 119, 131, 180, 314, 323, 344], Rough Set (Rough Graph) [288–293], Hyperfuzzy set[126,143,207,362], and Plithogenic Set (Plithogenic Graph) [121,132,338,339,357], are well-known in the literature. Future research could explore how these concepts behave when applied to Graph Neural Networks, Hypergraph Neural Networks, and SuperHyperGraph Neural Networks. Such investigations could also shed light on whether these extensions result in more efficient and effective networks. This area holds significant potential for advancing understanding and innovation.

Funding

This research received no external funding.

Acknowledgments

After publishing this paper as a preprint, we received valuable feedback on improving the manuscript from Dr. Azza A. Taha. We would like to take this opportunity to express our heartfelt gratitude.

We also humbly extend our deepest thanks to everyone who has provided invaluable support, enabling the successful completion of this work. Additionally, we sincerely appreciate all readers who have taken the time to engage with this study. Finally, we extend our utmost respect and gratitude to the authors of the references cited in this paper. Your significant contributions are greatly acknowledged.

Data Availability

This paper does not involve any data analysis.

Ethical Approval

This article does not involve any research with human participants or animals.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Disclaimer

This study primarily focuses on theoretical aspects, and its application to practical scenarios has not yet been validated. Future research may involve empirical testing and refinement of the proposed methods. The authors have made every effort to ensure that all references cited in this paper are accurate and appropriately attributed. However, unintentional errors or omissions may occur. The authors bear no legal responsibility for inaccuracies in external sources, and readers are encouraged to verify the information provided in the references independently. Furthermore, the interpretations and opinions expressed in this paper are solely those of the authors and do not necessarily reflect the views of any affiliated institutions.

References

[1] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 333–342, 2011.

[2] Mujahid Abbas, Ghulam Murtaza, and Florentin Smarandache. *Basic operations on hypersoft sets and hypersoft point*. Infinite Study, 2020.

[3] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

[4] Mohamed Abdel-Basset, Mai Mohamed, Mohamed Elhoseny, Le Hoang Son, Francisco Chiclana, and Abdel Nasser H. Zaied. Cosine similarity measures of bipolar neutrosophic set for diagnosis of bipolar disorder diseases. *Artificial intelligence in medicine*, 101:101735, 2019.

[5] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[6] Tero Aittokallio and Benno Schwikowski. Graph-based methods for analysing networks in cell biology. *Briefings in bioinformatics*, 7(3):243–255, 2006.

[7] D. Ajay, S. John Borg, and P. Chellamani. Domination in pythagorean neutrosophic graphs with an application in fuzzy intelligent decision making. In *International Conference on Intelligent and Fuzzy Systems*, pages 667–675, Cham, July 2022. Springer International Publishing.

[8] Muhammad Akram. Bipolar fuzzy graphs. *Information sciences*, 181(24):5548–5564, 2011.

[9] Muhammad Akram and Noura Omair Alshehri. Intuitionistic fuzzy cycles and intuitionistic fuzzy trees. *The Scientific World Journal*, 2014(1):305836, 2014.

[10] Muhammad Akram and Bijan Davvaz. Strong intuitionistic fuzzy graphs. *Filomat*, 26(1):177–196, 2012.

[11] Muhammad Akram and Wieslaw A. Dudek. Intuitionistic fuzzy hypergraphs with applications. *Inf. Sci.*, 218:182–193, 2013.

[12] Muhammad Akram, MG Karunambigai, K Palanivel, and S Sivasankar. Balanced bipolar fuzzy graphs. *Journal of advanced research in pure mathematics*, 6(4):58–71, 2014.

[13] Muhammad Akram and Anam Luqman. Bipolar neutrosophic hypergraphs with applications. *Journal of Intelligent & Fuzzy Systems*, 33(3):1699–1713, 2017.

[14] Muhammad Akram and Anam Luqman. Intuitionistic single-valued neutrosophic hypergraphs. *OPSEARCH*, 54:799 – 815, 2017.

[15] Muhammad Akram and Anam Luqman. Fuzzy hypergraphs and related extensions. In *Studies in Fuzziness and Soft Computing*, 2020.

[16] Muhammad Akram and Anam Luqman. Hypergraphs for interval-valued structures. In *Fuzzy Hypergraphs and Related Extensions*, pages 125–154. Springer, 2020.

[17] Muhammad Akram, Hafsa M Malik, Sundas Shahzadi, and Florentin Smarandache. Neutrosophic soft rough graphs with application. *Axioms*, 7(1):14, 2018.

[18] Muhammad Akram, Danish Saleem, and Talal Al-Hawary. Spherical fuzzy graphs with application to decision-making. *Mathematical and Computational Applications*, 25(1):8, 2020.

[19] Muhammad Akram, Sundas Shahzadi, and AB Saeid. Single-valued neutrosophic hypergraphs. *TWMS Journal of Applied and Engineering Mathematics*, 8(1):122–135, 2018.

[20] Qeethara Al-Shayea. Artificial neural networks in medical diagnosis. *International Journal of Research Publication and Reviews*, 2024.

[21] Md Tanvir Alam, Chowdhury Farhan Ahmed, Md Samiullah, and Carson K Leung. Mining frequent patterns from hypergraph databases. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 3–15. Springer, 2021.

[22] Md. Tanvir Alam, Chowdhury Farhan Ahmed, Md. Samiullah, and Carson Kai-Sang Leung. Mining frequent patterns from hypergraph databases. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2021.

[23] Eric Alcaide. Improving graph property prediction with generalized readout functions. *arXiv preprint arXiv:2009.09919*, 2020.

[24] Ebrahem Ateatullah Algehyne, Muhammad Lawan Jibril, Naseh A Algehainy, Osama Abdulaziz Alamri, and Abdullah Khaled J Alzahrani. Fuzzy neural network expert system with an improved gini index random forest-based feature importance measure algorithm for early diagnosis of breast cancer in saudi arabia. *Big Data Cogn. Comput.*, 6:13, 2022.

[25] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[26] Mohammed Alqahtani, M Kaviyarasu, Anas Al-Masarwah, and M Rajeshwari. Application of complex neutrosophic graphs in hospital infrastructure design. *Mathematics*, 12(5):719, 2024.

[27] Lilas Alrahis and Ozgur Sinanoglu. Graph neural networks for hardware vulnerability analysis-can you trust your gnn? In *2023 IEEE 41st VLSI Test Symposium (VTS)*, pages 1–4. IEEE, 2023.

[28] Mohammed Alshikho, Maissam Jdid, and Said Broumi. Artificial intelligence and neutrosophic machine learning in the diagnosis and detection of covid 19. *Journal Prospects for Applied Mathematics and Data Analysis*, 1(2), 2023.

[29] Farshad Andam, Ezzatollah Asgharizadeh, and Mohammadreza Taghizadeh-Yazdi. Designing a model for healthcare services supply chain performance evaluation using neutrosophic multiple attribute decision-making technique. *International Journal of Nonlinear Analysis and Applications*, 15(9):307–318, 2024.

[30] Christophe Andrieu, A. Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72, 2010.

[31] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.

[32] Howard Anton. Elementary linear algebra. 1970.

[33] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[34] Alejandro Barredo Arrieta, Natalia Díaz Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, A. Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Inf. Fusion*, 58:82–115, 2019.

[35] Cevdet Aykanat, B Barla Cambazoglu, and Bora Uçar. Multi-level direct k-way hypergraph partitioning with multiple constraints and fixed vertices. *Journal of Parallel and Distributed Computing*, 68(5):609–625, 2008.

[36] Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[37] Elisa C. Baek, Ryan Hyon, Karina López, Emily S. Finn, M. A. Porter, and Carolyn Parkinson. In-degree centrality in a social network is linked to coordinated neural activity. *Nature Communications*, 13, 2022.

[38] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[39] Wenhui Bai, Juanjuan Ding, and Chao Zhang. Dual hesitant fuzzy graphs with applications to multi-attribute decision making. *International Journal of Cognitive Computing in Engineering*, 1:18–26, 2020.

[40] Rassul Bairamkulov and Eby Friedman. Graphs in vlsi circuits and systems. In *Graphs in VLSI*, pages 59–100. Springer, 2022.

[41] Rassul Bairamkulov and Eby G Friedman. *Graphs in VLSI*. Springer, 2023.

[42] Alexandru T Balaban. Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, 25(3):334–343, 1985.

[43] Anuradha Banerjee, Basav Roychoudhury, and Bidyut Jyoti Gogoi. Determining rank in the market using a neutrosophic decision support system. *Journal of Business Analytics*, 3:138 – 157, 2020.

[44] Chaity Banerjee, Tathagata Mukherjee, and Eduardo Pasiliao Jr. The multi-phase relu activation function. In *Proceedings of the 2020 ACM Southeast Conference*, pages 239–242, 2020.

[45] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[46] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

[47] Veysi Başhan, Hakan Demirel, and Muhammet Gul. An fmea-based topsis approach under single valued neutrosophic sets for maritime risk evaluation: the case of ship navigation safety. *Soft Computing*, 24(24):18749–18764, 2020.

[48] Ilyes Batatia, D'avid P'eter Kov'acs, Gregor N. C. Simm, Christoph Ortner, and Gábor Csányi. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields. *ArXiv*, abs/2206.07697, 2022.

[49] Kornelia M. Batko and Andrzej lzak. The use of big data analytics in healthcare. *Journal of Big Data*, 9, 2022.

[50] G. Baudat and Fatiha Anouar. Feature vector selection and projection using kernels. *Neurocomputing*, 55:21–38, 2003.

[51] Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.

[52] Leonid S Bershtein and Alexander V Bozhenyuk. Fuzzy graphs and fuzzy hypergraphs. In *Encyclopedia of Artificial Intelligence*, pages 704–709. IGI Global, 2009.

[53] Anushree Bhattacharya and Madhumangal Pal. A fuzzy graph theory approach to the facility location problem: A case study in the indian banking system. *Mathematics*, 11(13):2992, 2023.

[54] Fanghui Bi, Tiantian He, Yuxuan Xie, and Xin Luo. Two-stream graph convolutional network-incorporated latent feature analysis. *IEEE Transactions on Services Computing*, 16:3027–3042, 2023.

[55] Jakub Binkowski, Albert Sawczyn, Denis Janiak, Piotr Bielak, and Tomasz Kajdanowicz. Graph-level representations using ensemble-based readout functions. In *International Conference on Computational Science*, pages 393–405. Springer, 2023.

[56] Pranab Biswas, Surapati Pramanik, and Bibhas Chandra Giri. Single valued bipolar pentapartitioned neutrosophic set and its application in madm strategy. 2022.

[57] Bela Bollobas. Modern graph theory. In *Graduate Texts in Mathematics*, 2002.

[58] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[59] Fateh Boutekkouk. Digital color image processing using intuitionistic fuzzy hypergraphs. *Int. J. Comput. Vis. Image Process.*, 11:21–40, 2021.

[60] Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 1, 2013.

[61] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[62] S Broumi and Tomasz Witczak. Heptapartitioned neutrosophic soft set. *International Journal of Neutrosophic Science*, 18(4):270–290, 2022.

[63] Said Broumi, Assia Bakali, Mohamed Talea, Florentin Smarandache, and V. Venkateswara Rao. Bipolar complex neutrosophic graphs of type 1. *viXra*, 2018.

[64] Said Broumi, Swaminathan Mohanaselvi, Tomasz Witczak, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Complex fermatean neutrosophic graph and application to decision making. *Decision Making: Applications in Management and Engineering*, 2023.

[65] Said Broumi, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Interval valued neutrosophic graphs. *Critical Review, XII*, 2016:5–33, 2016.

[66] A. Buck and James M. Keller. Evaluating path costs in multi-attributed fuzzy weighted graphs. *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2019.

[67] Samuel Rota Bulò and Marcello Pelillo. A game-theoretic approach to hypergraph clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1312–1327, 2009.

[68] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019.

[69] Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. Hypergraph structure learning for hypergraph neural networks. In *IJCAI*, pages 1923–1929, 2022.

[70] Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, linda Qiao, and Hongyan Li. Hypergraph structure learning for hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*, 2022.

[71] Yukun Cao and Yunfeng Li. An intelligent fuzzy-based recommendation system for consumer electronic products. *Expert Syst. Appl.*, 33:230–240, 2007.

[72] Matteo Carandini and David J Heeger. Normalization as a canonical neural computation. *Nature reviews neuroscience*, 13(1):51–62, 2012.

[73] Timoteo Carletti, Federico Battiston, Giulia Cencetti, and Duccio Fanelli. Random walks on hypergraphs. *Physical review E*, 101(2):022308, 2020.

[74] Timotéo Carletti, Duccio Fanelli, and Renaud Lambiotte. Random walks and community detection in hypergraphs. *Journal of Physics: Complexity*, 2, 2020.

[75] T-H Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. Spectral properties of hypergraph laplacian and approximation algorithms. *Journal of the ACM (JACM)*, 65(3):1–48, 2018.

[76] Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57(9):250, 2024.

[77] Chaofan Chen, Zelei Cheng, Zuotian Li, and Manyi Wang. Hypergraph attention networks. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1560–1565. IEEE, 2020.

[78] Haotian Chen and Jialiang Xie. Eeg-based tsk fuzzy graph neural network for driver drowsiness estimation. *Information Sciences*, 679:121101, 2024.

[79] Hsinchun Chen, Roger H. L. Chiang, and Veda C. Storey. Business intelligence and analytics: From big data to big impact. *MIS Q.*, 36:1165–1188, 2012.

[80] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 180–189, 2020.

[81] Cédric Chevalier and Ilya Safro. Comparison of coarsening schemes for multilevel graph partitioning. In *Learning and Intelligent Optimization*, 2009.

[82] Uthsav Chitra and Benjamin J. Raphael. Random walks on hypergraphs with edge-dependent vertex weights. In *International Conference on Machine Learning*, 2019.

[83] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. Reweighted random walks for graph matching. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part V 11*, pages 492–505. Springer, 2010.

[84] Vaughn Climenhaga. Markov chains and mixing times. 2013.

[85] Martina Contisciani, Federico Battiston, and Caterina De Bacco. Inference of hyperedges and overlapping communities in hypergraphs. *Nature communications*, 13(1):7229, 2022.

[86] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[87] Chris Cornelis, Pieter M. M. De Kesel, and Etienne E. Kerre. Shortest paths in fuzzy weighted graphs. *International Journal of Intelligent Systems*, 19, 2004.

[88] Bruce A Craig and Peter P Sendi. Estimation of the transition matrix of a discrete-time markov chain. *Health economics*, 11(1):33–42, 2002.

[89] Ganqu Cui, Yufeng Du, Cheng Yang, Jie Zhou, Liang Xu, Xing Zhou, Xingyi Cheng, and Zhiyuan Liu. Evaluating modules in graph contrastive learning. *arXiv preprint arXiv:2106.08171*, 2021.

[90] Qionghai Dai and Yue Gao. Mathematical foundations of hypergraph. In *Hypergraph Computation*, pages 19–40. Springer, 2023.

[91] Suman Das, Rakhal Das, and Surapati Pramanik. Single valued pentapartitioned neutrosophic graphs. *Neutrosophic Sets and Systems*, 50(1):225–238, 2022.

[92] Suman Das, Rakhal Das, and Binod Chandra Tripathy. Topology on rough pentapartitioned neutrosophic set. *Iraqi Journal of Science*, 2022.

[93] Danilo Dell'Agnello, Anna Maria Fanelli, Corrado Mencar, and Massimo Minervini. Serendipitous fuzzy item recommendation with profilematcher. In *International Workshop on Fuzzy Logic and Applications*, 2011.

[94] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *AAAI Conference on Artificial Intelligence*, 2021.

[95] Narsingh Deo. *Graph theory with applications to engineering and computer science*. Courier Dover Publications, 2016.

[96] Chinthaka Sajith Devinda and Anil Kumar. Application of fuzzy machine learning algorithm in agro-geography. 2020.

[97] Keith J. Devlin. Fundamentals of contemporary set theory. 1979.

[98] P. M. Dhanya, A. Sreekumar, M. Jathavedan, and P. B. Ramkumar. Algebra of morphological dilation on intuitionistic fuzzy hypergraphs. *International journal of scientific research in science, engineering and technology*, 4:300–308, 2018.

[99] P. M. Dhanya, A. Sreekumar, M. Jathavedan, and P. B. Ramkumar. On constructing morphological erosion of intuitionistic fuzzy hypergraphs. *The Journal of Analysis*, 27:583 – 603, 2018.

[100] Reinhard Diestel. Graduate texts in mathematics: Graph theory.

[101] Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173(33):12, 2005.

[102] Reinhard Diestel. *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks), 2024.

[103] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be more with less: Hypergraph attention networks for inductive text classification. *arXiv preprint arXiv:2011.00387*, 2020.

[104] Boyu Du, Jingya Zhou, Ling Liu, and Xiaolong She. Fl-gnn: Efficient fusion of fuzzy neural network and graph neural network. In *ECAI 2024*, pages 1768–1775. IOS Press, 2024.

[105] Aurélien Ducournau and Alain Bretto. Random walks in directed hypergraphs and application to semi-supervised image segmentation. *Computer Vision and Image Understanding*, 120:91–102, 2014.

[106] Zdenek Dvorák, Archontia C. Giannopoulou, and Dimitrios M. Thilikos. Forbidden graphs for tree-depth. *Eur. J. Comb.*, 33:969–979, 2012.

[107] Philip Ehrlich. *Real numbers, generalizations of the reals, and theories of continua*, volume 242. Springer Science & Business Media, 2013.

[108] Reem Essameldin, Ahmed A. Ismail, and Saad Mohamed Darwish. Quantifying opinion strength: A neutrosophic inference system for smart sentiment analysis of social media network. *Applied Sciences*, 2022.

[109] Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201, 2009.

[110] Shimon Even. *Graph algorithms*. Cambridge University Press, 2011.

[111] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[112] Juntao Fei and Lunhaojie Liu. Real-time nonlinear model predictive control of active power filter using self-feedback recurrent fuzzy neural network estimator. *IEEE Transactions on Industrial Electronics*, 69:8366–8376, 2022.

[113] Juntao Fei, Zhe Wang, Xiao Liang, Zhilin Feng, and Yuncan Xue. Fractional sliding-mode control for microgyroscope based on multilayer recurrent fuzzy neural network. *IEEE Transactions on Fuzzy Systems*, 30:1712–1721, 2022.

[114] Song Feng, Emily Heath, Brett Jefferson, Cliff Joslyn, Henry Kvinge, Hugh D Mitchell, Brenda Praggastis, Amie J Eisfeld, Amy C Sims, Larissa B Thackray, et al. Hypergraph models of biological networks to identify genes critical to pathogenic viral response. *BMC bioinformatics*, 22(1):287, 2021.

[115] Yifan Feng, Haoxuan You, Zizhao Zhang, R. Ji, and Yue Gao. Hypergraph neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

[116] Alessio Ferone and Alfredo Petrosino. A neuro fuzzy approach for handling structured data. In *Scalable Uncertainty Management*, 2008.

[117] Ronald C. Freiwald. An introduction to set theory and topology. 2014.

[118] Dongqi Fu and Jingrui He. Sdg: A simplified and dynamic graph neural network. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2273–2277, 2021.

[119] Takaaki Fujita. Note for hypersoft filter and fuzzy hypersoft filter. *Multicriteria Algorithms With Applications*, 5:32–51, 2024.

[120] Takaaki Fujita. Note for neutrosophic incidence and threshold graph. *SciNexuses*, 1:97–125, 2024.

[121] Takaaki Fujita. A review of the hierarchy of plithogenic, neutrosophic, and fuzzy graphs: Survey and applications. *ResearchGate(Preprint)*, 2024.

[122] Takaaki Fujita. Short note of supertree-width and n-superhypertree-width. *Neutrosophic Sets and Systems*, 77:54–78, 2024.

[123] Takaaki Fujita. Survey of intersection graphs, fuzzy graphs and neutrosophic graphs. *ResearchGate*, July 2024.

[124] Takaaki Fujita. Survey of planar and outerplanar graphs in fuzzy and neutrosophic graphs. *ResearchGate*, July 2024.

[125] Takaaki Fujita. Survey of trees, forests, and paths in fuzzy and neutrosophic graphs. July 2024.

[126] Takaaki Fujita. *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond*. Biblio Publishing, 2025.

[127] Takaaki Fujita. A comprehensive discussion on fuzzy hypersoft expert, superhypersoft, and indetermsoft graphs. *Neutrosophic Sets and Systems*, 77:241–263, 2025.

[128] Takaaki Fujita. Fundamental computational problems and algorithms for superhypergraphs. March 2025.

[129] Takaaki Fujita and Florentin Smarandache. Antipodal turiyam neutrosophic graphs. *Neutrosophic Optimization and Intelligent Systems*, 5:1–13, 2024.

[130] Takaaki Fujita and Florentin Smarandache. A concise study of some superhypergraph classes. *Neutrosophic Sets and Systems*, 77:548–593, 2024.

[131] Takaaki Fujita and Florentin Smarandache. A short note for hypersoft rough graphs. *HyperSoft Set Methods in Engineering*, 3:1–25, 2024.

[132] Takaaki Fujita and Florentin Smarandache. Study for general plithogenic soft expert graphs. *Plithogenic Logic and Computation*, 2:107–121, 2024.

[133] Takaaki Fujita and Florentin Smarandache. Uncertain automata and uncertain graph grammar. *Neutrosophic Sets and Systems*, 74:128–191, 2024.

[134] Takaaki Fujita and Florentin Smarandache. Neutrosophic circular-arc graphs and proper circular-arc graphs. *Neutrosophic Sets and Systems*, 78:1–30, 2025.

[135] Francois Le Gall. Faster algorithms for rectangular matrix multiplication. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 514–523, 2012.

[136] A Nagoor Gani and K Radha. On regular fuzzy graphs. 2008.

[137] Shenghua Gao, Ivor Wai-Hung Tsang, and Liang-Tien Chia. Laplacian sparse coding, hypergraph laplacian sparse coding, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):92–104, 2012.

[138] Weihao Gao and Tao Mo. Hypergraph clustering with inhomogeneous partitions of hyperedges pan. 2017.

[139] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3181–3199, 2022.

[140] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2548–2566, 2020.

[141] Fayed F. M. Ghaleb, Azza A. Taha, Maryam Hazman, Mahmoud Abd Ellatif, and Mona Abbass. On quasi cycles in hypergraph databases. *IEEE Access*, 8:147560–147568, 2020.

[142] Meysam Gheisarnejad, Ardashir Mohammadzadeh, Hamed Farsizadeh, and Mohammad Hassan Khooban. Stabilization of 5g telecom converter-based deep type-3 fuzzy machine learning control for telecom applications. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69:544–548, 2022.

[143] Jayanta Ghosh and Tapas Kumar Samanta. Hyperfuzzy sets and hyperfuzzy group. *Int. J. Adv. Sci. Technol*, 41:27–37, 2012.

[144] Puspendu Giri, Somnath Paul, and Bijoy Krishna Debnath. A fuzzy graph theory and matrix approach (fuzzy gtma) to select the best renewable energy alternative in india. *Applied Energy*, 358:122582, 2024.

[145] S Gomathy, D Nagarajan, S Broumi, and M Lathamaheswari. *Plithogenic sets and their application in decision making*. Infinite Study, 2020.

[146] Zengtai Gong and Junhu Wang. Hesitant fuzzy graphs, hesitant fuzzy hypergraphs and fuzzy graph decisions. *Journal of Intelligent & Fuzzy Systems*, 40(1):865–875, 2021.

[147] Bahareh Goodarzi, Farzad Khorasani, Vivek Sarkar, and Dhrubajyoti Goswami. High performance multilevel graph partitioning on gpu. *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pages 769–778, 2019.

[148] Sathyanarayanan Gopalakrishnan, Supriya Sridharan, Soumya Ranjan Nayak, Janmenjoy Nayak, and Swaminathan Venkataraman. Central hubs prediction for bio networks by directed hypergraph-ga with validation to covid-19 ppi. *Pattern Recognition Letters*, 153:246–253, 2022.

[149] Igor I Gorban. Hyper-random phenomena: definition and description. *Information Theories and Applications*, 15(3):203–211, 2008.

[150] Igor I Gorban. *Randomness and Hyper-randomness*. Springer, 2018.

[151] II Gorban. The hyperrandom functions and their description. *Radioelectronics and Communications Systems*, 49(1):1–9, 2006.

[152] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 21–32, 1999.

[153] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: A survey. In *Mathematical Foundations of Computer Science 2001: 26th International Symposium, MFCS 2001 Mariánské Lázne, Czech Republic, August 27–31, 2001 Proceedings 26*, pages 37–57. Springer, 2001.

[154] Georg Gottlob and Reinhard Pichler. Hypergraphs in model checking: Acyclicity and hypertree-width versus clique-width. *SIAM Journal on Computing*, 33(2):351–378, 2004.

[155] Linfeng Gou and Yu Zhong. A new fault diagnosis method based on attributes weighted neutrosophic set. *IEEE Access*, 7:117740–117748, 2019.

[156] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[157] Peter J. Green. Markov chain monte carlo in practice. 1996.

[158] Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018.

[159] Mingyu Guan, Anand Padmanabha Iyer, and Taesoo Kim. Dynagraph: dynamic graph neural networks at scale. In *Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, pages 1–10, 2022.

[160] Abhishek Guleria and Rakesh Kumar Bajaj. T-spherical fuzzy graphs: Operations and applications in various selection processes. *Arabian Journal for Science and Engineering*, 45:2177 – 2193, 2019.

[161] Muhammad Gulistan, Naveed Yaqoob, Zunaira Rashid, Florentin Smarandache, and Hafiz Abdul Wahab. A study on neutrosophic cubic graphs with real life applications in industries. *Symmetry*, 10(6):203, 2018.

[162] Xinyu Guo, Bingjie Tian, and Xuedong Tian. Hfgnn-proto: Hesitant fuzzy graph neural network-based prototypical network for few-shot text classification. *Electronics*, 11(15):2423, 2022.

[163] Zhiwei Guo, Keping Yu, Alireza Jolfaei, Gang Li, Feng Ding, and Amin Beheshti. Mixed graph neural network-based fake news detection for sustainable vehicular social networks. *IEEE Transactions on Intelligent Transportation Systems*, 24(12):15486–15498, 2022.

[164] Tom Gur, Noam Lifshitz, and Siqi Liu. Hypercontractivity on high dimensional expanders. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 176–184, 2022.

[165] Venkatesan Guruswami and Sai Sandeep. An algorithmic study of the hypergraph turán problem. *ArXiv*, abs/2008.07344, 2020.

[166] Dae Geun Ha, Tae Wook Ha, Junghyuk Seo, and Myoung-Ho Kim. Index-based searching for isomorphic subgraphs in hypergraph databases. *Journal of KIISE*, 2019.

[167] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[168] M Hamidi and A Borumand Saeid. Accessible single-valued neutrosophic graphs. *Journal of Applied Mathematics and Computing*, 57:121–146, 2018.

[169] Mohammad Hamidi and Marzieh Rahmati. On binary decision hypertree (hyperdiagram). *AUT Journal of Mathematics and Computing*, 5(2):117–130, 2024.

[170] Mohammad Hamidi, Florentin Smarandache, and Mohadeseh Taghinezhad. *Decision Making Based on Valued Fuzzy Superhypergraphs*. Infinite Study, 2023.

[171] Mohammad Hamidi and Mohadeseh Taghinezhad. *Application of Superhypergraphs-Based Domination Number in Real World*. Infinite Study, 2023.

[172] Jianlong Hao, Zhibin Liu, Qiwei Sun, Chen Zhang, and Jie Wang. A static-dynamic hypergraph neural network framework based on residual learning for stock recommendation. *Complex.*, 2024:5791802:1–5791802:12, 2024.

[173] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[174] Koby Hayashi, Sinan G Aksoy, Cheong Hee Park, and Haesun Park. Hypergraph random walks, laplacians, and clustering. In *Proceedings of the 29th acm international conference on information & knowledge management*, pages 495–504, 2020.

[175] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*, 2018.

[176] Wei He and Yiting Dong. Adaptive fuzzy neural network control for a constrained robot using impedance learning. *IEEE Transactions on Neural Networks and Learning Systems*, 29:1174–1186, 2018.

[177] Yixuan He, Quan Gan, David Wipf, Gesine D Reinert, Junchi Yan, and Mihai Cucuringu. Gnnrank: Learning global rankings from pairwise comparisons via directed graph neural networks. In *international conference on machine learning*, pages 8581–8612. PMLR, 2022.

[178] Yixuan He, Gesine Reinert, David Wipf, and Mihai Cucuringu. Robust angular synchronization via directed graph neural networks. *arXiv preprint arXiv:2310.05842*, 2023.

[179] Yixuan He, Xitong Zhang, Junjie Huang, Benedek Rozemberczki, Mihai Cucuringu, and Gesine Reinert. Pytorch geometric signed directed: a software package on graph neural networks for signed and directed graphs. In *Learning on Graphs Conference*, pages 12–1. PMLR, 2024.

[180] R. Hema, R. Sudharani, and M. Kavitha. A novel approach on plithogenic interval valued neutrosophic hypersoft sets and its application in decision making. *Indian Journal Of Science And Technology*, 2023.

[181] Nasimeh Heydaribeni, Xinrui Zhan, Ruisi Zhang, Tina Eliassi-Rad, and Farinaz Koushanfar. Hypop: Distributed constrained combinatorial optimization leveraging hypergraph neural networks. *ArXiv*, abs/2311.09375, 2023.

[182] Karel Hrbacek and Thomas Jech. Introduction to set theory, revised and expanded. 2017.

[183] Chao Hu, Ruishi Yu, Binqi Zeng, Yu Zhan, Ying Fu, Quan Zhang, Rongkai Liu, and Heyuan Shi. Hyperattack: Multi-gradient-guided white-box adversarial structure attack of hypergraph neural networks. *ArXiv*, abs/2302.12407, 2023.

[184] Shenglong Hu and Liqun Qi. Algebraic connectivity of an even uniform hypergraph. *Journal of Combinatorial Optimization*, 24:564–579, 2012.

[185] Shenglong Hu and Liqun Qi. The laplacian of a uniform hypergraph. *Journal of Combinatorial Optimization*, 29:331–366, 2015.

[186] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[187] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021.

[188] Liangsong Huang, Yu Hu, Yuxia Li, PK Kishore Kumar, Dipak Koley, and Arindam Dey. A study of regular and irregular neutrosophic graphs with real life applications. *Mathematics*, 7(6):551, 2019.

[189] S Satham Hussain, N Durga, Muhammad Aslam, G Muhiuddin, and Ganesh Ghorai. New concepts on quadripartitioned neutrosophic competition graph with application. *International Journal of Applied and Computational Mathematics*, 10(2):57, 2024.

[190] S Satham Hussain, N Durga, Rahmonlou Hossein, and Ghorai Ganesh. New concepts on quadripartitioned single-valued neutrosophic graph with real-life application. *International Journal of Fuzzy Systems*, 24(3):1515–1529, 2022.

[191] S Satham Hussain, Hossein Rashmonlou, R Jahir Hussain, Sankar Sahoo, Said Broumi, et al. Quadripartitioned neutrosophic graph structures. *Neutrosophic Sets and Systems*, 51(1):17, 2022.

[192] S Satham Hussain, Isnaini Rosyida, Hossein Rashmanlou, and Farshid Mofidnakhaei. Interval intuitionistic neutrosophic sets with its applications to interval intuitionistic neutrosophic graphs and climatic analysis. *Computational and Applied Mathematics*, 40(4):121, 2021.

[193] Satham Hussain, Jahir Hussain, Isnaini Rosyida, and Said Broumi. Quadripartitioned neutrosophic soft graphs. In *Handbook of Research on Advances and Applications of Fuzzy Sets and Logic*, pages 771–795. IGI Global, 2022.

[194] Yasmine M Ibrahim, Reem Essameldin, and Saad M Darwish. An adaptive hate speech detection approach using neutrosophic neural networks for social media forensics. *Computers, Materials & Continua*, 79(1), 2024.

[195] Borislav Iordanov. Hypergraphdb: a generalized graph database. In *Web-Age Information Management: WAIM 2010 International Workshops: IWGD 2010, XMLDM 2010, WCMT 2010, Jiuzhaigou Valley, China, July 15-17, 2010 Revised Selected Papers 11*, pages 25–36. Springer, 2010.

[196] Alex Gabriel Lara Jacome, Elizabeth Mayorga Aldaz, Miguel Ramos Argilagos, and Darvin Manuel Ramírez Guerra. Neutrosophic perspectives in healthcare decision-making: Navigating complexity with ethics, information, and collaboration. *Neutrosophic Sets and Systems*, 62(1):15, 2023.

[197] Sirus Jahanpanah and Roohallah Daneshpayeh. On derived superhyper be-algebras. *Neutrosophic Sets and Systems*, 57(1):21, 2023.

[198] Sirus Jahanpanah and Roohallah Daneshpayeh. An outspread on valued logic superhyperalgebras. *Facta Universitatis, Series: Mathematics and Informatics*, 2024.

[199] Chiranjibe Jana, Tapan Senapati, Monoranjan Bhowmik, and Madhumangal Pal. On intuitionistic fuzzy g-subalgebras of g-algebras. *Fuzzy Information and Engineering*, 7(2):195–209, 2015.

[200] Wasnaa Kadhim Jawad and Abbas M. Al-Bakry. Big data analytics: A survey. *Iraqi Journal for Computers and Informatics*, 2022.

[201] Thomas Jech. *Set theory: The third millennium edition, revised and expanded*. Springer, 2003.

[202] Janice Jeffs and Benoît Mario Papillon. Globalization, the new economy and project management: a graph theory perspective. *The Journal of Modern Project Management*, 7, 2019.

[203] Bukyoung Jhun, Minjae Jo, and Byungnam Kahng. Simplicial sis model in scale-free uniform hypergraph. *Journal of Statistical Mechanics: Theory and Experiment*, 2019, 2019.

[204] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*, 2019.

[205] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *ArXiv*, abs/2101.11174, 2021.

[206] Hayoung Jo and Seong-Whan Lee. Edge conditional node update graph neural network for multi-variate time series anomaly detection. *Information Sciences*, page 121062, 2024.

[207] Young Bae Jun, Kul Hur, and Kyoung Ja Lee. Hyperfuzzy subalgebras of bck/bci-algebras. *Annals of Fuzzy Mathematics and Informatics*, 2017.

[208] Ilanthenral Kandasamy, WB Vasantha, Jagan M Obbineni, and Florentin Smarandache. Sentiment analysis of tweets using refined neutrosophic sets. *Computers in Industry*, 115:103180, 2020.

[209] Vasantha Kandasamy, K Ilanthenral, and Florentin Smarandache. *Neutrosophic graphs: a new dimension to graph theory*. Infinite Study, 2015.

[210] Xiaojun Kang, Xinchuan Li, Hong Yao, Dan Li, Bo Jiang, Xiaoyue Peng, Tiejun Wu, Shihua Qi, and Lijun Dong. Dynamic hypergraph neural networks based on key hyperedges. *Inf. Sci.*, 616:37–51, 2022.

[211] Komal Kapoor, Dhruv Sharma, and Jaideep Srivastava. Weighted node degree centrality for hypergraphs. *2013 IEEE 2nd Network Science Workshop (NSW)*, pages 152–155, 2013.

[212] Abdullah Kargın and Memet Şahin. Superhyper groups and neutro–superhyper groups. *2023 Neutrosophic SuperHyperAlgebra And New Types of Topologies*, page 25, 2023.

[213] Abdullah Kargın, Florentin Smarandache, and Memet Şahin. *New Type Hyper Groups, New Type SuperHyper Groups and Neutro-New Type SuperHyper Groups*. Infinite Study, 2023.

[214] George Karypis. Multilevel hypergraph partitioning. In *Multilevel Optimization in VLSICAD*, pages 125–154. Springer, 2003.

[215] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. In *Proceedings of the 34th annual Design Automation Conference*, pages 526–529, 1997.

[216] George Karypis and Vipin Kumar. Analysis of multilevel graph partitioning. *Proceedings of the IEEE/ACM SC95 Conference*, pages 29–29, 1995.

[217] George Karypis and Vipin Kumar. Multilevel graph partitioning schemes. In *International Conference on Parallel Processing*, 1995.

[218] George Karypis and Vipin Kumar. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th annual ACM/IEEE design automation conference*, pages 343–348, 1999.

[219] Peter Keevash. Hypergraph turan problems. *Surveys in combinatorics*, 392:83–140, 2011.

[220] Tamás Képes. The critical node detection problem in hypergraphs using weighted node degree centrality. *PeerJ Computer Science*, 9, 2023.

[221] Huda E Khalid, Gonca Durmaz Gungor, and Muslim A Noah. *Neutrosophic SuperHyper Bi-Topological Spaces: Extra Topics*. Infinite Study, 2024.

[222] Eun-Sol Kim, Woo Young Kang, Kyoung-Woon On, Yu-Jung Heo, and Byoung-Tak Zhang. Hypergraph attention networks for multimodal learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14581–14590, 2020.

[223] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[224] Dalibor Krleža and Krešimir Fertalj. Graph matching using hierarchical fuzzy graph neural networks. *Ieee transactions on fuzzy systems*, 25(4):892–904, 2016.

[225] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

[226] Adarsh Kumar, C. P. Gandhi, Yuqing Zhou, He sheng Tang, and Jiawei Xiang. Fault diagnosis of rolling element bearing based on symmetric cross entropy of neutrosophic sets. *Measurement*, 2020.

[227] Marius Leordeanu and Cristian Sminchisescu. Efficient hypergraph clustering. In *International Conference on Artificial Intelligence and Statistics*, 2012.

[228] Juanhui Li, Harry Shomer, Jiayu Ding, Yiqi Wang, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Are message passing neural networks really helpful for knowledge graph completion? In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[229] Kunhao Li, Zhenhua Huang, and Zhaohong Jia. Rahg: A role-aware hypergraph neural network for node classification in graphs. *IEEE Transactions on Network Science and Engineering*, 10:2098–2108, 2023.

[230] Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. *ArXiv*, abs/1709.01249, 2017.

[231] Wei Li, Bin Xiang, Fan Yang, Yuchen Rong, Yanbin Yin, Jianhua Yao, and Han Zhang. scmhnn: a novel hypergraph neural network for integrative analysis of single-cell epigenomic, transcriptomic and proteomic data. *Briefings in bioinformatics*, 24 6, 2023.

[232] Xiaowei Liao, Yong Xu, and Haibin Ling. Hypergraph neural networks for hypergraph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1266–1275, 2021.

[233] Shinyoung Lim, Kwanyong Lee, Okhwan Byeon, and Taiyun Kim. Efficient iris recognition through improvement of feature vector and classifier. *ETRI Journal*, 23, 2001.

[234] Guifang Lin and Wei Shen. Research on convolutional neural network based on improved relu piecewise activation function. *Procedia computer science*, 131:977–984, 2018.

[235] Sheng-Wei Lin and Huai-Wei Lo. An fmea model for risk assessment of university sustainability: using a combined itara with topsis-al approach based neutrosophic sets. *Annals of Operations Research*, pages 1–27, 2023.

[236] Dayun Liu, Xianghui Li, Liangliang Zhang, Xiaowen Hu, Jiaxuan Zhang, Zhirong Liu, and Lei Deng. Hgnnlda: Predicting lncrna-drug sensitivity associations via a dual channel hypergraph neural network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20:3547–3555, 2023.

[237] Huaiyuan Liu, Donghua Yang, Xianzhang Liu, Xinglei Chen, Zhiyu Liang, Hongzhi Wang, Yong Cui, and Jun Gu. Todynet: temporal dynamic graph neural network for multivariate time series classification. *Information Sciences*, page 120914, 2024.

[238] Li Liu and Fanzhang Li. A survey on dynamic fuzzy machine learning. *ACM Computing Surveys*, 55:1 – 42, 2022.

[239] Luotao Liu, Feng Huang, Xuan Liu, Zhankun Xiong, Menglu Li, Congzhi Song, and Wen Zhang. Multi-view contrastive learning hypergraph neural network for drug-microbe-disease association prediction. In *International Joint Conference on Artificial Intelligence*, 2023.

[240] Shengyuan Liu, Pei Lv, Yuzhen Zhang, Jie Fu, Junjin Cheng, Wanqing Li, Bing Zhou, and Mingliang Xu. Semi-dynamic hypergraph neural network for 3d pose estimation. In *International Joint Conference on Artificial Intelligence*, 2020.

[241] Xizhi Liu and Dhruv Mubayi. A hypergraph turán problem with no stability. *Combinatorica*, 42:433–462, 2019.

[242] Yu-Ting Liu, Yang-Yin Lin, Shang-Lin Wu, Chun-Hsiang Chuang, and Chin-Teng Lin. Brain dynamics in predicting driving fatigue using a recurrent self-evolving fuzzy neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 27:347–360, 2016.

[243] Yue Liu, Wenxuan Tu, Sihang Zhou, Xinwang Liu, Linxuan Song, Xihong Yang, and En Zhu. Deep graph clustering via dual correlation reduction. In *AAAI Conference on Artificial Intelligence*, 2021.

[244] Zijian Liu, Yang Luo, Xitong Pu, Geyong Min, and Chunbo Luo. A multi-modal hypergraph neural network via parametric filtering and feature sampling. *IEEE Transactions on Big Data*, 9:1365–1379, 2023.

[245] Han Lu, Quanxue Gao, Qianqian Wang, Ming Yang, and Wei Xia. Centerless multi-view k-means based on the adjacency matrix. In *AAAI Conference on Artificial Intelligence*, 2023.

[246] Jie Lu, Guangzhi Ma, and Guangquan Zhang. Fuzzy machine learning: A comprehensive framework and systematic review. *IEEE Transactions on Fuzzy Systems*, 32:3861–3878, 2024.

[247] Xiaoyi Luo, Jiaheng Peng, and Jun Liang. Directed hypergraph attention network for traffic forecasting. *IET Intelligent Transport Systems*, 16(1):85–98, 2022.

[248] Anam Luqman, Muhammad Akram, and Florentin Smarandache. Complex neutrosophic hypergraphs: New social network models. *Algorithms*, 12:234, 2019.

[249] Anam Luqman, Muhammad Akram, and Florentin Smarandache. Complex neutrosophic hypergraphs: new social network models. *Algorithms*, 12(11):234, 2019.

[250] Xuejiao Ma, Yu Jin, and Qingli Dong. A generalized dynamic fuzzy neural network based on singular spectrum analysis optimized by brain storm optimization for short-term wind speed forecasting. *Appl. Soft Comput.*, 54:296–312, 2017.

[251] Zhongtian Ma, Zhiguo Jiang, and Haopeng Zhang. Hyperspectral image classification using feature fusion hypergraph convolution neural network. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2021.

[252] Rupkumar Mahapatra, Sovan Samanta, Madhumangal Pal, Tofigh Allahviranloo, and Antonios Kalampakas. A study on linguistic z-graph and its application in social networks. *Mathematics*, 12(18):2898, 2024.

[253] Rupkumar Mahapatra, Sovan Samanta, Madhumangal Pal, and Qin Xin. Link prediction in social networks by neutrosophic graph. *Int. J. Comput. Intell. Syst.*, 13:1699–1713, 2020.

[254] Pradip Kumar Maji, Ranjit Biswas, and A Ranjan Roy. Soft set theory. *Computers & mathematics with applications*, 45(4-5):555–562, 2003.

[255] Muhammad Aslam Malik, Ali Hassan, Said Broumi, Assia Bakali, Mohamed Talea, and Florentin Smarandache. Isomorphism of bipolar single valued neutrosophic hypergraphs. *Collected Papers. Volume IX: On Neutrosophic Theory and Its Applications in Algebra*, page 72, 2022.

[256] Rama Mallick and Surapati Pramanik. Pentapartitioned neutrosophic set and its properties. *Neutrosophic Sets and Systems*, 35:49, 2020.

[257] J. Manyika. Big data: The next frontier for innovation, competition, and productivity. 2011.

[258] Nivetha Martin and Florentin Smarandache. Concentric plithogenic hypergraph based on plithogenic hypersoft sets ? a novel outlook. *Neutrosophic Sets and Systems*, 33:5, 2020.

[259] Sunil Mathew and MS Sunitha. Cycle connectivity in weighted graphs. *Proyecciones (Antofagasta)*, 30(1):1–17, 2011.

[260] Keith R. Matthews. Elementary linear algebra. 1998.

[261] Justin J Miller. Graph database applications and concepts with neo4j. In *Proceedings of the southern association for information systems conference, Atlanta, GA, USA*, volume 2324, pages 141–147, 2013.

[262] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018.

[263] Mai Mohamed, Mohamed Abdel-Basset, Abdel-Nasser Hussien, and Florentin Smarandache. Using neutrosophic sets to obtain pert three-times estimates in project management. *Infinite Study*, 2017.

[264] Mai Mohamed and Asmaa Elsayed. A novel multi-criteria decision making approach based on bipolar neutrosophic set for evaluating financial markets in egypt. *Multicriteria Algorithms with Applications*, 2024.

[265] Mona Mohamed, Alaa Elmor, Florentin Smarandache, and Ahmed A Metwaly. An efficient superhypersoft framework for evaluating llms-based secure blockchain platforms. *Neutrosophic Sets and Systems*, 72:1–21, 2024.

[266] Dmitriy Molodtsov. Soft set theory-first results. *Computers & mathematics with applications*, 37(4-5):19–31, 1999.

[267] John N Mordeson and Sunil Mathew. *Advanced topics in fuzzy graph theory*, volume 375. Springer, 2019.

[268] John N Mordeson and Premchand S Nair. *Fuzzy graphs and fuzzy hypergraphs*, volume 46. Physica, 2012.

[269] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[270] Norzieha Mustapha, Suriana Alias, Roliza Md Yasin, Ilyani Abdullah, and Said Broumi. Cardiovascular diseases risk analysis using distance-based similarity measure of neutrosophic sets. 2021.

[271] M Myvizhi, Ahmed M Ali, Ahmed Abdelhafeez, and Haitham Rizk Fadlallah. *MADM Strategy Application of Bipolar Single Valued Heptapartitioned Neutrosophic Set*. Infinite Study, 2023.

[272] S Narasimman, M Shanmugapriya, R Sundareswaran, Laxmi Rathour, Lakshmi Narayan Mishra, Vinita Dewangan, and Vishnu Narayan Mishra. Identification of influential factors affecting student performance in semester examinations in the educational institution using score topological indices in single valued neutrosophic graphs. *Neutrosophic Sets and Systems*, 75:224–240, 2025.

[273] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[274] TM Nishad, Talal Ali Al-Hawary, and B Mohamed Harif. General fuzzy graphs. *Ratio Mathematica*, 47, 2023.

[275] Ya-Wei Niu, Cun-Quan Qu, Guang-Hui Wang, and Gui-Ying Yan. Rwhmda: random walk on hypergraph for microbe-disease association prediction. *Frontiers in microbiology*, 10:1578, 2019.

[276] Oluseyi Olurotimi, Amir Dembo, and Thomas Kailath. Neural network weight matrix synthesis using optimal control techniques. In *Neural Information Processing Systems*, 1989.

[277] Alessandro Pagano, Raffaele Giordano, and Ivan Portoghese. A pipe ranking method for water distribution network resilience assessment based on graph-theory metrics aggregated through bayesian belief networks. *Water Resources Management*, 36(13):5091–5106, 2022.

[278] Sunay P Pai and Rajesh S Prabhu Gaonkar. Modelling uncertainty using neutrosophic sets for precise risk assessment of marine systems. *International Journal of System Assurance Engineering and Management*, pages 1–8, 2023.

[279] Madhumangal Pal, Sovan Samanta, and Ganesh Ghorai. *Modern trends in fuzzy graph theory*. Springer, 2020.

[280] Erlin Pan and Zhao Kang. Multi-view contrastive graph clustering. In *Neural Information Processing Systems*, 2021.

[281] Sakshi Dev Pandey, AS Ranadive, and Sovan Samanta. Bipolar-valued hesitant fuzzy graph and its application. *Social Network Analysis and Mining*, 12(1):14, 2022.

[282] Jiahao Pang and Gene Cheung. Graph laplacian regularization for image denoising: Analysis in the continuous domain. *IEEE Transactions on Image Processing*, 26:1770–1785, 2016.

[283] Christos H Papadimitriou. Computational complexity. In *Encyclopedia of computer science*, pages 260–265. 2003.

[284] R Parvathi, S Thilagavathi, and MG Karunambigai. Intuitionistic fuzzy hypergraphs. *Cybernetics and Information Technologies*, 9(2):46–53, 2009.

[285] Rangasamy Parvathi, S. Thilagavathi, and M. G. Karunambigai. Operations on intuitionistic fuzzy hypergraphs. *International Journal of Computer Applications*, 51:46–54, 2012.

[286] T Pathinathan, J Jon Arockiaraj, and J Jesintha Rosline. Hesitancy fuzzy graphs. *Indian Journal of Science and Technology*, 8(35):1–5, 2015.

[287] Vasile Patrascu. Penta and hexa valued representation of neutrosophic information. *arXiv preprint arXiv:1603.03729*, 2016.

[288] Zdzisław Pawlak. Rough sets. *International journal of computer & information sciences*, 11:341–356, 1982.

[289] Zdzislaw Pawlak. Rough set theory and its applications to data analysis. *Cybernetics & Systems*, 29(7):661–688, 1998.

[290] Zdzisław Pawlak. Rough sets and intelligent data analysis. *Information sciences*, 147(1-4):1–12, 2002.

[291] Zdzislaw Pawlak, Jerzy Grzymala-Busse, Roman Slowinski, and Wojciech Ziarko. Rough sets. *Communications of the ACM*, 38(11):88–95, 1995.

[292] Zdzislaw Pawlak, Lech Polkowski, and Andrzej Skowron. Rough set theory. *KI*, 15(3):38–39, 2001.

[293] Zdzislaw Pawlak, S. K. Michael Wong, Wojciech Ziarko, et al. Rough sets: probabilistic versus deterministic approach. *International Journal of Man-Machine Studies*, 29(1):81–95, 1988.

[294] Nicole Pearcy, Jonathan J Crofts, and Nadia Chuzhanova. Hypergraph models of metabolism. *International Journal of Biological, Veterinary, Agricultural and Food Engineering*, 8(8):752–756, 2014.

[295] Pramod Kumar Poladi and K. Sagar. Reinforcement learning and neuro-fuzzy gnn-based vertical handover decision on internet of vehicles. *Concurrency and Computation: Practice and Experience*, 35, 2023.

[296] Stephen Pryke. Towards a social network theory of project governance. *Construction Management and Economics*, 23:927 – 939, 2005.

[297] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[298] Shio Gai Quek, Ganeshsree Selvachandran, D Ajay, P Chellamani, David Taniar, Hamido Fujita, Phet Duong, Le Hoang Son, and Nguyen Long Giang. New concepts of pentapartitioned neutrosophic graphs and applications for determining safest paths and towns in response to covid-19. *Computational and Applied Mathematics*, 41(4):151, 2022.

[299] Marzieh Rahmati and Mohammad Hamidi. Extension of g-algebras to superhyper g-algebras. *Neutrosophic Sets and Systems*, 55(1):34, 2023.

[300] Marzieh Rahmati and Mohammad Hamidi. On strong super hyper eq algebras: A proof-of-principle study. *Plithogenic Logic and Computation*, 2:29–36, 2024.

[301] Sajad Ramezani, Mauzama Firdaus, and Lili Mou. Claim-centric and sentiment guided graph attention network for rumour detection. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3235–3241, 2024.

[302] M Ramya, Sandesh Murali, and R.Radha. Bipolar quadripartitioned neutrosophic soft set. 2022.

[303] Alfréd Rényi. Representations for real numbers and their ergodic properties. *Acta Mathematica Academiae Scientiarum Hungarica*, 8:477–493, 1957.

[304] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*, pages 287–297. Springer, 2008.

[305] Soufiane Rital, H. Cherifi, and Serge Miguet. K-way hypergraph partitioning and color image segmentation. 2006.

[306] Azriel Rosenfeld. Fuzzy graphs. In *Fuzzy sets and their applications to cognitive and decision processes*, pages 77–95. Elsevier, 1975.

[307] Kashob Kumar Roy, Amit Roy, AKM Mahbubur Rahman, M Ashraful Amin, and Amin Ahsan Ali. Structure-aware hierarchical graph pooling using information bottleneck. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[308] Toni Sagayaraj and Carsten Eickhoff. Image-like graph representations for improved molecular property prediction. *arXiv preprint arXiv:2111.10695*, 2021.

[309] A. A. Salama, Haitham A. El-Ghareeb, Ayman M. Manie, and Momen M Lotfy. Utilizing neutrosophic set in social network analysis e-learning systems. 2014.

[310] AA Salama, A Haitham, A Manie, and M Lotfy. Utilizing neutrosophic set in social network analysis e-learning systems. *International Journal of Information Science and Intelligent System*, 3(2):61–72, 2014.

[311] Sovan Samanta and Madhumangal Pal. Bipolar fuzzy hypergraphs. *International Journal of Fuzzy Logic Systems*, 2(1):17–28, 2012.

[312] Jimena Montes De Oca Sánchez, Myriam Paulina Barreno Sánchez, Miriam Janeth Pantoja Burbano, and Osmanys Pérez Peña. Neutrosophic marketing strategy and consumer behavior. *Neutrosophic Sets and Systems*, 62:209–216, 2023.

[313] S Satham Hussain, Durga Nagarajan, Hossein Rashmanlou, and Farshid Mofidnakhaei. Novel supply chain decision making model under m-polar quadripartitioned neutrosophic environment. *Journal of Applied Mathematics and Computing*, pages 1–26, 2024.

[314] P Sathya, Nivetha Martin, and Florentine Smarandache. Plithogenic forest hypersoft sets in plithogenic contradiction based multi-criteria decision making. *Neutrosophic Sets and Systems*, 73:668–693, 2024.

[315] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, Tyler Derr, and Rajiv Ratn Shah. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 497–504, 2021.

[316] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[317] Sebastian Schlag, Vitali Henne, Tobias Heuer, Henning Meyerhenke, Peter Sanders, and Christian Schulz. k-way hypergraph partitioning via n-level recursive bisection. *ArXiv*, abs/1511.03137, 2015.

[318] Idan Schwartz, Seunghak Yu, Tamir Hazan, and Alexander G Schwing. Factor graph attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2039–2048, 2019.

[319] John Scott. *What is social network analysis?* Bloomsbury Academic, 2012.

[320] Robert Sedgewick and Kevin Wayne. *Algorithms*. Addison-wesley professional, 2011.

[321] Ranu Sewada, Ashwani Jangid, Piyush Kumar, and Neha Mishra. Explainable artificial intelligence (xai). *international journal of food and nutritional sciences*, 2023.

[322] Gulfam Shahzadi, Muhammad Akram, Arsham Borumand Saeid, et al. An application of single-valued neutrosophic sets in medical diagnosis. *Neutrosophic sets and systems*, 18:80–88, 2017.

[323] Francina Shalini. Trigonometric similarity measures of pythagorean neutrosophic hypersoft sets. *Neutrosophic Systems with Applications*, 2023.

[324] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *ArXiv*, abs/1811.05868, 2018.

[325] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7912–7921, 2019.

[326] Lilian Shi. Correlation coefficient of simplified neutrosophic sets for bearing fault diagnosis. *Shock and Vibration*, 2016:1–11, 2016.

[327] Xiaolong Shi, Saeed Kosari, Hossein Rashmanlou, Said Broumi, and S Satham Hussain. Properties of interval-valued quadripartitioned neutrosophic graphs with real-life application. *Journal of Intelligent & Fuzzy Systems*, 44(5):7683–7697, 2023.

[328] Pritpal Singh. A neutrosophic-entropy based clustering algorithm (nebca) with hsv color system: A special application in segmentation of parkinson's disease (pd) mr images. *Computer methods and programs in biomedicine*, 189:105317, 2020.

[329] S Sivasankar and Said Broumi. A new algorithm to determine the density of a balanced neutrosophic graph and its application to enhance education quality. In *Handbook of research on the applications of neutrosophic sets theory and their extensions in education*, pages 1–17. IGI Global, 2023.

[330] Damjan Škulj. Discrete time markov chains with interval probabilities. *International journal of approximate reasoning*, 50(8):1314–1329, 2009.

[331] F. Smarandache. Introduction to superhyperalgebra and neutrosophic superhyperalgebra. *Journal of Algebraic Hyperstructures and Logical Algebras*, 2022.

[332] Florentin Smarandache. A unifying field in logics: Neutrosophic logic. In *Philosophy*, pages 1–141. American Research Press, 1999.

[333] Florentin Smarandache. *Definitions derived from neutrosophics*. Infinite Study, 2003.

[334] Florentin Smarandache. Neutrosophic set-a generalization of the intuitionistic fuzzy set. *International journal of pure and applied mathematics*, 24(3):287, 2005.

[335] Florentin Smarandache. *A unifying field in logics: neutrosophic logic. Neutrosophy, neutrosophic set, neutrosophic probability: neutrsophic logic. Neutrosophy, neutrosophic set, neutrosophic probability*. Infinite Study, 2005.

[336] Florentin Smarandache. Neutrosophic physics: More problems, more solutions. 2010.

[337] Florentin Smarandache. n-valued refined neutrosophic logic and its applications to physics. *Infinite study*, 4:143–146, 2013.

[338] Florentin Smarandache. *Plithogenic set, an extension of crisp, fuzzy, intuitionistic fuzzy, and neutrosophic sets-revisited*. Infinite study, 2018.

[339] Florentin Smarandache. Plithogeny, plithogenic set, logic, probability, and statistics. *arXiv preprint arXiv:1808.03948*, 2018.

[340] Florentin Smarandache. n-superhypergraph and plithogenic n-superhypergraph. *Nidus Idearum*, 7:107–113, 2019.

[341] Florentin Smarandache. *Extension of HyperGraph to n-SuperHyperGraph and to Plithogenic n-SuperHyperGraph, and Extension of HyperAlgebra to n-ary (Classical-/Neutro-/Anti-) HyperAlgebra*. Infinite Study, 2020.

[342] Florentin Smarandache. History of superhyperalgebra and neutrosophic superhyperalgebra (revisited again). *Neutrosophic Algebraic Structures and Their Applications*, page 10, 2022.

[343] Florentin Smarandache. *Introduction to the n-SuperHyperGraph-the most general form of graph today*. Infinite Study, 2022.

[344] Florentin Smarandache. *Practical applications of IndetermSoft Set and IndetermHyperSoft Set and introduction to TreeSoft Set as an extension of the MultiSoft Set*. Infinite Study, 2022.

[345] Florentin Smarandache. *The SuperHyperFunction and the Neutrosophic SuperHyperFunction (revisited again)*, volume 3. Infinite Study, 2022.

[346] Florentin Smarandache. Decision making based on valued fuzzy superhypergraphs. 2023.

[347] Florentin Smarandache. Foundation of the superhypersoft set and the fuzzy extension superhypersoft set: A new vision. *Neutrosophic Systems with Applications*, 11:48–51, 2023.

[348] Florentin Smarandache. New types of topologies and neutrosophic topologies. *Neutrosophic Systems with Applications*, 1:1–3, 2023.

[349] Florentin Smarandache. New types of topologies and neutrosophic topologies (improved version). *Neutrosophic Sets and Systems*, 57(1):14, 2023.

[350] Florentin Smarandache. *SuperHyperFunction, SuperHyperStructure, Neutrosophic SuperHyperFunction and Neutrosophic SuperHyperStructure: Current understanding and future directions*. Infinite Study, 2023.

[351] Florentin Smarandache. Foundation of superhyperstructure & neutrosophic superhyperstructure. *Neutrosophic Sets and Systems*, 63(1):21, 2024.

[352] Florentin Smarandache. Superhyperstructure & neutrosophic superhyperstructure, 2024. Accessed: 2024-12-01.

[353] Florentin Smarandache. Short introduction to standard and nonstandard neutrosophic set and logic. *Neutrosophic Sets and Systems*, 77:395–404, 2025.

[354] Florentin Smarandache and Said Broumi. *Neutrosophic graph theory and algorithms*. IGI Global, 2019.

[355] Florentin Smarandache and NM Gallup. Generalization of the intuitionistic fuzzy set to the neutrosophic set. In *International Conference on Granular Computing*, pages 8–42. Citeseer, 2006.

[356] Florentin Smarandache, WB Kandasamy, and K Ilanthenral. Applications of bimatrices to some fuzzy and neutrosophic models. 2005.

[357] Florentin Smarandache and Nivetha Martin. *Plithogenic n-super hypergraph in novel multi-attribute decision making*. Infinite Study, 2020.

[358] Florentin Smarandache, Memet Şahin, Derya Bakbak, Vakkas Uluçay, and Abdullah Kargın. *Neutrosophic SuperHyperAlgebra and New Types of Topologies*. Infinite Study, 2023.

[359] Florentin Smarandache and AA Salama. Neutrosophic crisp set theory. 2015.

[360] Florentin Smarandache, A. Saranya, A. Kalavathi, and S. Krishnaprakash. Neutrosophic superhypersoft sets. *Neutrosophic Sets and Systems*, 77:41–53, 2024.

[361] Chenguang Song, Yiyang Teng, Yangfu Zhu, Siqi Wei, and Bin Wu. Dynamic graph neural network for fake news detection. *Neurocomputing*, 505:362–374, 2022.

[362] Seok-Zun Song, Seon Jeong Kim, and Young Bae Jun. Hyperfuzzy ideals in bck/bci-algebras. *Mathematics*, 5(4):81, 2017.

[363] Francesco Sorrentino. Synchronization of hypernetworks of coupled dynamical systems. *New Journal of Physics*, 14(3):033035, 2012.

[364] Fazeelat Sultana, Muhammad Gulistan, Mumtaz Ali, Naveed Yaqoob, Muhammad Khan, Tabasam Rashid, and Tauseef Ahmed. A study of plithogenic graphs: applications in spreading coronavirus disease (covid-19) globally. *Journal of ambient intelligence and humanized computing*, 14(10):13139–13159, 2023.

[365] Jinjun Tang, Fang Liu, Wenhui Zhang, Ruimin Ke, and Yajie Zou. Lane-changes prediction based on adaptive fuzzy neural network. *Expert Syst. Appl.*, 91:452–463, 2018.

[366] Jinjun Tang, Fang Liu, Yajie Zou, Weibin Zhang, and Yinhai Wang. An improved fuzzy neural network for traffic speed prediction considering periodic characteristic. *IEEE Transactions on Intelligent Transportation Systems*, 18:2340–2350, 2017.

[367] Trevor Tarr. *Leibniz, Calculus, and The Hyperreal Numbers*. PhD thesis, 2024.

[368] Sérgio Dinis teixeira de Sousa, Isabel Lopes, and Eusébio Nunes. Graph theory approach to quantify uncertainty of performance measures. 2015.

[369] Lev Telyatnikov, Maria Sofia Bucarelli, Guillermo Bernardez, Olga Zaghen, Simone Scardapane, and Pietro Lió. Hypergraph neural networks through the lens of message passing: A common perspective to homophily and architecture design. *ArXiv*, abs/2310.07684, 2023.

[370] Ankit Thakkar and Kinjal Chaudhari. Predicting stock trend using an integrated term frequency-inverse document frequency-based feature weight matrix with neural networks. *Appl. Soft Comput.*, 96:106684, 2020.

[371] P. Thirunavukarasu and R. Suresh. Annals of on regular complex neutrosophic graphs. 2017.

[372] P. Thirunavukarasu and R. Suresh. On regular complex neutrosophic graphs. 2017.

[373] Yuanyuan Tian, Richard C Mceachin, Carlos Santos, David J States, and Jignesh M Patel. Saga: a subgraph matching tool for biological graphs. *Bioinformatics*, 23(2):232–239, 2007.

[374] Eric J. Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature Medicine*, 25:44 – 56, 2019.

[375] Vicenç Torra. Hesitant fuzzy sets. *International journal of intelligent systems*, 25(6):529–539, 2010.

[376] Vicenç Torra and Yasuo Narukawa. On hesitant fuzzy sets and decision. In *2009 IEEE international conference on fuzzy systems*, pages 1378–1382. IEEE, 2009.

[377] Mirko Torrisi, Gianluca Pollastri, and Quan Le. Deep learning methods in protein structure prediction. *Computational and Structural Biotechnology Journal*, 18:1301–1310, 2020.

[378] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[379] Aleksandar Trifunovic and William J Knottenbelt. A parallel algorithm for multilevel k-way hypergraph partitioning. In *Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, pages 114–121. IEEE, 2004.

[380] Nenad Trinajstic. *Chemical graph theory*. CRC press, 2018.

[381] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *ArXiv*, abs/2006.16904, 2020.

[382] Tran Manh Tuan, Pham Minh Chuan, Mumtaz Ali, Tran Thi Ngan, Mamta Mittal, and Le Hoang Son. Fuzzy and neutrosophic modeling for link prediction in social networks. *Evolving Systems*, 10:629 – 634, 2018.

[383] Vakkas Ulucay and Memet Sahin. Intuitionistic fuzzy soft expert graphs with application. *Uncertainty Discourse and Applications*, 1(1):1–10, 2024.

[384] D Ulyanov. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[385] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[386] Guillaume Verdon, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary. Quantum graph neural networks. *arXiv preprint arXiv:1909.12264*, 2019.

[387] Arvind Kumar Verma and Sunil Rajotia. Feature vector: a graph-based feature recognition methodology. *International Journal of Production Research*, 42:3219 – 3234, 2004.

[388] Johannes Von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[389] George Voutsadakis. Introduction to set theory. *A Problem Based Journey from Elementary Number Theory to an Introduction to Matrix Theory*, 2021.

[390] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *European Conference on Computer Vision*, 2018.

[391] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering: A deep attentional embedding approach. In *International Joint Conference on Artificial Intelligence*, 2019.

[392] Dan Wang, Xi ke Tian, Lu Li, Chao Yang, Langchun Xing, and Jun Shang. A fault diagnosis algorithm for distribution networks based on graph convolutional neural networks. *2024 20th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 1–5, 2024.

[393] Jia Wang and Zhenyuan Wang. Using neural networks to determine sugeno measures by statistics. *Neural Networks*, 10:183–195, 1997.

[394] Jianling Wang, Kaize Ding, Ziwei Zhu, and James Caverlee. Session-based recommendation with hypergraph attention networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pages 82–90. SIAM, 2021.

[395] Jingcheng Wang, Yong Zhang, Yun Wei, Yongli Hu, Xinglin Piao, and Baocai Yin. Metro passenger flow prediction via dynamic hypergraph convolution networks. *IEEE Transactions on Intelligent Transportation Systems*, 22:7891–7903, 2021.

[396] Qian Wang and Zengtai Gong. An application of fuzzy hypergraphs and hypergraphs in granular computing. *Inf. Sci.*, 429:296–314, 2018.

[397] Qian Wang and Zengtai Gong. Structural centrality in fuzzy social networks based on fuzzy hypergraph theory. *Computational and Mathematical Organization Theory*, 26:236 – 254, 2020.

[398] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 950–958, 2019.

[399] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[400] Yuge Wang, Xibei Yang, Qiguo Sun, Yuhua Qian, and Qihang Guo. Purity skeleton dynamic hypergraph neural network. *Neurocomputing*, 610:128539, 2024.

[401] Yuxin Wang, Quan Gan, Xipeng Qiu, Xuanjing Huang, and David Paul Wipf. From hypergraph energy functions to hypergraph neural networks. In *International Conference on Machine Learning*, 2023.

[402] Stanley Wasserman and Katherine Faust. Social network analysis: Methods and applications. 1994.

[403] Ramon Elias Weber, Caitlin Mueller, and Christoph Reinhart. A hypergraph model shows the carbon reduction potential of effective space use in housing. *Nature Communications*, 15(1):8327, 2024.

[404] Tong Wei, Junlin Hou, and Rui Feng. Fuzzy graph neural network for few-shot learning. In *2020 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2020.

[405] Yuxiang Wei, Shiqi Wang, and Yun Li. Graph theory based machine learning for analog circuit design. In *2023 28th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE, 2023.

[406] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[407] Tomasz Witczak. Interior and closure in anti-minimal and anti-biminimal spaces in the frame of anti-topology. *Neutrosophic Sets and Systems*, 56(1):29, 2023.

[408] Wolfgang Woess. *Random walks on infinite graphs and groups*. Number 138. Cambridge university press, 2000.

[409] Dianshuang Wu, Guangquan Zhang, and Jie Lu. A fuzzy preference tree-based recommender system for personalized business-to-business e-services. *IEEE Transactions on Fuzzy Systems*, 23:29–43, 2015.

[410] Hongjie Wu, Weizhong Lu, Meiling Qian, Yu Zhang, Yijie Ding, Jiawei Shen, Xiaoyi Chen, Haiou Li, and Qiming Fu. Identification of membrane protein types based using hypergraph neural network. *Current Bioinformatics*, 2023.

[411] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495, 2017.

[412] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[413] Ru xia Liang, Qian Zhang, and Jianqiang Wang. Hierarchical fuzzy graph attention network for group recommendation. *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2021.

[414] Jun Xie, Qiguang Miao, Ruyi Liu, Wentian Xin, Lei Tang, Sheng Zhong, and Xuesong Gao. Attention adjacency matrix based graph convolutional networks for skeleton-based action recognition. *Neurocomputing*, 440:230–239, 2021.

[415] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[416] Xiaolong Xu, Qinting Jiang, Peiming Zhang, Xuefei Cao, Mohammad Hossein Khosravi, Linss T. Alex, Lianyong Qi, and Wanchun Dou. Game theory for distributed iov task offloading with fuzzy neural network in edge computing. *IEEE Transactions on Fuzzy Systems*, 30:4593–4604, 2022.

[417] Zeshui Xu. *Hesitant fuzzy sets theory*, volume 314. Springer, 2014.

[418] Yong Bo Xuan, Chang Qiang Huang, and Wang Xi Li. Air combat situation assessment by gray fuzzy bayesian network. *Applied Mechanics and Materials*, 69:114–119, 2011.

[419] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[420] Naveed Yaqoob and Muhammad Akram. *Complex neutrosophic graphs*. Infinite Study, 2018.

[421] Ouyang Yi, Bin Guo, Xing Tang, Xiuqiang He, Jian Xiong, and Zhiwen Yu. Learning cross-domain representation with multi-graph neural network. *ArXiv*, abs/1905.10095, 2019.

[422] Pairote Yiarayong. On 2-superhyperleftalmostsemihyp regroups. *Neutrosophic Sets and Systems*, 51(1):33, 2022.

[423] G George Yin and Qing Zhang. *Discrete-time Markov chains: two-time-scale methods and applications*, volume 55. Springer Science & Business Media, 2005.

[424] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. Local higher-order graph clustering. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[425] Nan Yin, Li Shen, Huan Xiong, Bin Gu, Chong Chen, Xian-Sheng Hua, Siwei Liu, and Xiao Luo. Messages are never propagated alone: Collaborative hypergraph neural network for time-series forecasting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:2333–2347, 2023.

[426] Zhizhuo Yin, Kai Han, Pengzi Wang, and Xi Zhu. H3gnn: Hybrid hierarchical hypergraph neural network for personalized session-based recommendation. *ACM Transactions on Information Systems*, 42:1 – 30, 2023.

[427] Jean-Gabriel Young, Giovanni Petri, and Tiago P Peixoto. Hypergraph reconstruction from network data. *Communications Physics*, 4(1):135, 2021.

[428] Jiajun Yu, Zhihao Wu, Jinyu Cai, Adele Lu Jia, and Jicong Fan. Kernel readout for graph neural networks.

[429] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5782–5799, 2022.

[430] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[431] Lotfi A Zadeh. A fuzzy-set-theoretic interpretation of linguistic hedges. 1972.

[432] Lotfi A Zadeh. Fuzzy sets and their application to pattern classification and clustering analysis. In *Classification and clustering*, pages 251–299. Elsevier, 1977.

[433] Lotfi A Zadeh. Fuzzy sets versus probability. *Proceedings of the IEEE*, 68(3):421–421, 1980.

[434] Lotfi A Zadeh. Fuzzy logic, neural networks, and soft computing. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 775–782. World Scientific, 1996.

[435] Lotfi A Zadeh. Fuzzy sets and information granularity. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 433–448. World Scientific, 1996.

[436] Lotfi A Zadeh. A note on prototype theory and fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: Selected papers by Lotfi A Zadeh*, pages 587–593. World Scientific, 1996.

[437] Lotfi Asker Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1(1):3–28, 1978.

[438] Jin Zeng, Gene Cheung, Michael K. Ng, Jiahao Pang, and Cheng Yang. 3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model. *IEEE Transactions on Image Processing*, 29:3474–3489, 2018.

[439] Chun-Yang Zhang, Yue-Na Lin, C. L. Philip Chen, Hong-Yu Yao, Hai-Chun Cai, and Wu-Peng Fang. Fuzzy representation learning on graph. *IEEE Transactions on Fuzzy Systems*, 31:3358–3370, 2023.

[440] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and N. Chawla. Heterogeneous graph neural network. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[441] Junlong Zhang and Yubin Luo. Degree centrality, betweenness centrality, and closeness centrality in social network. 2017.

[442] Leijie Zhang, Ye Shi, Yu-Cheng Chang, and Chin-Teng Lin. Hierarchical fuzzy neural networks with privacy preservation for heterogeneous big data. *IEEE Transactions on Fuzzy Systems*, 29(1):46–58, 2020.

[443] Yuanzhao Zhang, Maxime Lucas, and Federico Battiston. Higher-order interactions shape collective dynamics differently in hypergraphs and simplicial complexes. *Nature communications*, 14(1):1605, 2023.

[444] Yuhao Zhang, Peng Qi, and Christopher D. Manning. Graph convolution over pruned dependency trees improves relation extraction. *ArXiv*, abs/1809.10185, 2018.

[445] Hua Zhao, Zeshui Xu, Shousheng Liu, and Zhong Wang. Intuitionistic fuzzy mst clustering algorithms. *Computers & Industrial Engineering*, 62(4):1130–1140, 2012.

[446] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21:3848–3858, 2018.

[447] Wufan Zhao, Claudio Persello, and Alfred Stein. Extracting planar roof structures from very high resolution images using graph neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 187:34–45, 2022.

[448] Chunhang Zheng and Kechao Cai. Genet: A graph neural network-based anti-noise task-oriented semantic communication paradigm. *arXiv preprint arXiv:2403.18296*, 2024.

[449] Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S Yu, and Shirui Pan. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

[450] Guo Zhenyu and Zhang Wanhong. An efficient inference schema for gene regulatory networks using directed graph neural networks. In *2023 42nd Chinese Control Conference (CCC)*, pages 6829–6834. IEEE, 2023.

[451] Luying Zhong, Jinbin Yang, Zhaoliang Chen, and Shiping Wang. Contrastive graph convolutional networks with generative adjacency matrix. *IEEE Transactions on Signal Processing*, 71:772–785, 2023.

[452] Hongliang Zhou and Rik Sarkar. Leveraging graph machine learning for moonlighting protein prediction: A ppi network and physiochemical feature approach. *bioRxiv*, pages 2023–11, 2023.

[453] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[454] Peng Zhou, Zongqian Wu, Xiangxiang Zeng, Guoqiu Wen, Junbo Ma, and Xiaofeng Zhu. Totally dynamic hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*, 2023.

[455] Hao Zhu and Piotr Koniusz. Simple spectral graph convolution. In *International Conference on Learning Representations*, 2021.